quattor

Quattor Installation and User Guide

Version 1.1.x

EDMS id:	**Id**
Date:	**February 9, 2006**
Partner(s):	**UAM,CERN,INFN,IN2P3,NIKHEF**
Lead Partner:	**CERN**
Document status:	**DRAFT/RFC**
Author(s):	Rafael A. García Leiva (angel.leiva@uam.es) German Cancio Meliá (German.Cancio@cern.ch) Charles Loomis (Charles.Loomis@cern.ch) Piotr Poznański (Piotr.Poznanski@cern.ch) Marco Emilio Poleggi (Marco.Poleggi@cern.ch) Marc Poulhies (Marc.Poulhies@cern.ch) Stijn De Weirdt (sdeweird@vub.ac.be)
File:	**quattor-install-guide**

Abstract: *This document describes the installation and configuration of the quattor tool suite. Also it includes a short introduction to quattor operation.*

CONTENTS

1. INTRODUCTION

Quattor [1] is a system administration tool suite providing a powerful, portable and modular tool for the automated installation, configuration and management of machines running UNIX derivatives like Linux and Solaris. Quattor was started in the scope of the European DataGrid project (2001-2003) (see [1]); but development and maintenance continues to be coordinated by CERN (IT department), in collaboration with other partner institutes (in particular, UAM, INFN, IN2P3 and NIKHEF).

Quattor components are being used in production at the CERN computer centre, managing more than 2400 nodes (CPU worker nodes, tape and disk servers, infrastructure servers) on several Linux and Solaris platforms. Many other institutes and universities are using quattor to manage their own network infrastructures as well.

For more information about quattor, and to freely download the latest software release, check the quattor web page at `http://quattor.org`.

1.1. OBJECTIVES OF THIS DOCUMENT

This guide provides instructions to install the quattor tool suite, and a short introduction to quattor operation. This document covers a broad range of quattor environments, from small sites installation (less than one hundred client nodes), to very large sites installations (potentially thousands of nodes). This guide describes the installation of quattor on Scientific Linux 3, but similar procedures apply to other versions of RedHat Linux, and other Unix flavors (see Section 2.2. for a list of supported platforms, and Appendix A for the installation release notes of each platform).

This document is intended for experienced Unix system managers, but it does not assume any knowledge of the quattor architecture or terminology. For more information about the quattor tool suite, check the links under the '*documentation*' tab at quattor web page.

Help is appreciated!

The Quattor toolsuite is the result of a collaborative effort. If you have suggestions, comments, improvements on this installation guide please don't hesitate to contact us. If you have CVS access, it is possible to access the LaTeXsources of this document at `CVS:/elfms/quattor/documentation/installation-guide`. Feel free to send us patches!

1.2. APPLICATION AREA

This document applies to the quattor tool suite (see `http://quattor.org`) versions 1.1.x.

[1] quattor stands for **QU**attor is an **A**dministration **T**oolki**T** for **O**ptimizing **R**esources

1.3. ACRONYMS AND ABBREVIATIONS

The quattor-specific acronyms and abbreviations used in this guide are the following:

AII: Automated Installation Infrastructure

CAF: Common Application Framework

CCM: Configuration Cache Manager

CDB: Configuration Database

HLD: High Level Description

HLDL: High Level Description Language

NCM: Node Configuration Manager

NVA-API: Node View Access API

LLD: Low Level Description

LLDL: Low Level Description Language

NBP: Network Bootstrap Program

SPM: Software Package Management

SPMA: Software Package Management Agent

SWRep: Software Repository

REFERENCES

[1] *EDG: European DataGrid Project* http://www.edg.org

[2] *Quattor Quick Installation Guide*, version 1.0, Rafael García Leiva, Germán Cancio and Piotr Poznański.

[3] *Configuration Database Global Design*, version 2.0, Lionel Cons and Piotr Poznański.

[4] *CDB Synchronization Notification System Global Design*, version 1.1, Rafael García Leiva and Piotr Poznański.

[5] *SPM Software Repository Specification*, version 0.5, Andrey Kiryanov and Lev Shamardin.

[6] *Configuration Client Cache Specification*, version 2.0.0, Lionel Cons and Piotr Poznański.

[7] *Node View Access API Specification*, version 3.1.0, Piotr Poznański.

[8] *The NVA-API Tutorial*, Rafael A. García Leiva.

[9] *Node Configuration Manager Design*, version 0.92, Germán Cancio.

[10] *Software Package Management and Distribution Design*, version 0.1, Germán Cancio, Ian Neilson and Andrey Kiryanov.

[11] *Automated Installation Infrastructure Design*, version 0.2, Cristina Aiftimiei and Enrico Ferro.

[12] *Internet Systems Consortium (ISC) Dynamic Host Configuration Protocol (DHCP)*, http://www.isc.org/index.pl?/sw/dhcp

[13] *SysLinux Project*: http://syslinux.zytor.com/

[14] *Quattor LCG PEB Working Group*: https://svn.lal.in2p3.fr/LCG/QWG/web/index.html

[15] *Apache 2.0 SSL/TLS Strong Encryption FAQ*
`http://httpd.apache.org/docs-2.0/ssl/ssl_faq.html`

[16] *mod_SSL source code* `http://www.modssl.org/source/`

[17] *mod_SSL reference documentation*
`http://httpd.apache.org/docs-2.0/mod/mod_ssl.html`

[18] *mod_rewrite reference documentation*
`http://httpd.apache.org/docs-2.0/mod/mod_rewrite.html`

[19] *mod_proxy reference documentation*
`http://httpd.apache.org/docs-2.0/mod/mod_proxy.html`

[20] *mod_cache reference documentation*
`http://httpd.apache.org/docs-2.0/mod/mod_cache.html`

2. OVERVIEW OF QUATTOR INSTALLATION

2.1. QUATTOR ARCHITECTURE

The quattor information model is based on the distinction between the *desired configuration state* and the *actual configuration state* of nodes. The desired state is registered in a fabric-wide *Configuration Database* (CDB). A specially designed configuration language called 'Pan' is used for expressing and validating configurations, which can be composed out of reusable hierarchical building blocks called *templates*. A set of *Server Modules* support various query patterns of the information stored in CDB (currently SQL queries). Once validated, configurations are propagated to *Configuration Cache Manager* agents (CCM) running on client nodes and cached locally. Figure 1 shows the flow of the configuration information in quattor.

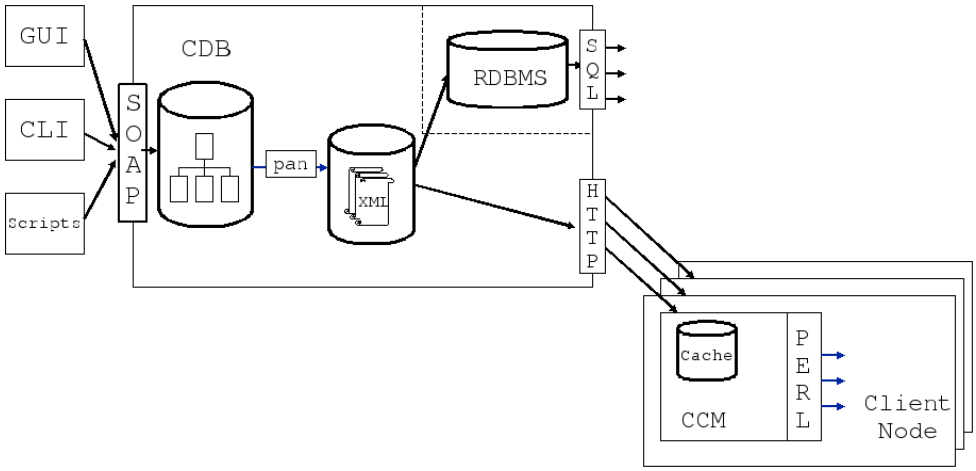

Figure 1: Configuration Information Management

Subsystems running on client nodes, called *configuration components*, take care of managing software packages and configuring local services. The *Software Package Management Agent* (SPMA) handles local software installations. The *Node Configuration Manager* (NCM) subsystem configures and reconfigures local systems. A *Software Repository* (SWRep) allows system managers to create, and remotely control (add, remove, query, etc), a repository storing all software packages used to install and manage client nodes. A subsystem called *Automated Installation Infrastructure* (AII) deal with the initial node installations, by providing drivers for generating installation time information like DHCP tables, installer control files (eg. KickStart or JumpStart), and user front-end tools. Figure 2 shows how the configuration information is deployed on client nodes.

For more information about quattor architecture, please refer to the quattor web page, under the *documentation* tab.

2.2. PREREQUISITES

A minimal installation of quattor needs at least a server node running Scientific Linux 3.

All the quattor software packages can be freely downloaded from the quattor web page. However, quattor uses many other services for its operation. Among these services, we can mention the basic servers: Web, DNS, CVS and SSH. And for the automatic installation of client nodes you need DHCP, TFTP and PXELinux. All those dependencies are mentioned in this installation guide when they are needed.

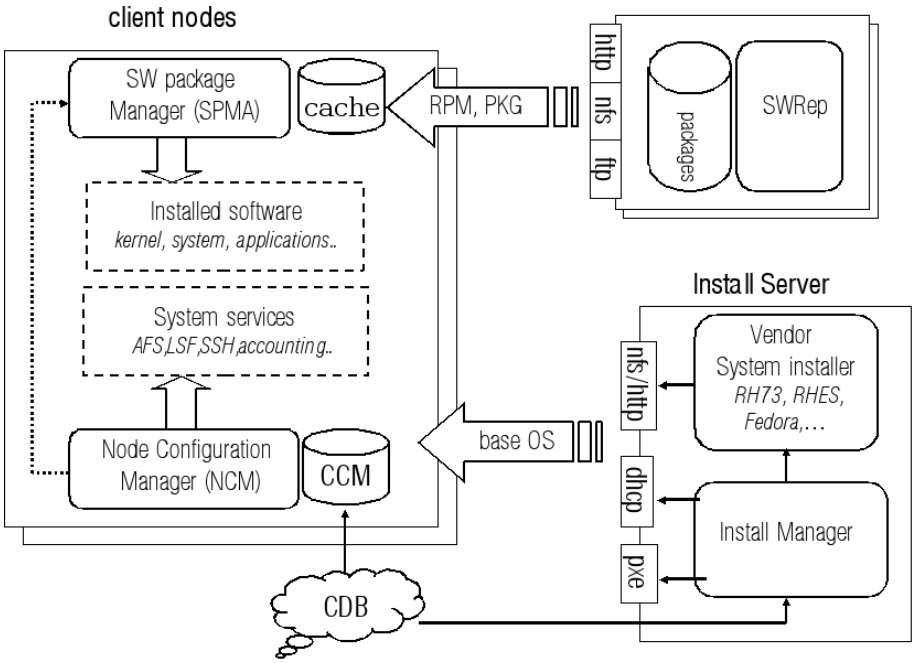

Figure 2: Configuration Deployment

OS	Version	Platform	Status
RedHat	7.3	i386	Supported
RedHat Enterprise Server	3	i386, IA64, x86_64	Supported
Scientific Linux	3	i386, IA64, x86_64	Supported
Solaris	9	Sun	Contact us

Table 1: Supported Platforms

Supported Platforms

The primary platform for quattor is Scientific Linux, but platform independence is a design principle. Quattor has been tested in the platforms described on Table 1. However, it may work in other platforms as well. If you have successfully installed and used quattor in any other platform, we would like to hear about it. Note, that specific packages for each platform are provided.

The examples and explanations provided in this guide assume either a RedHat ES3 or a Scientific Linux 3 installation. Similar procedures should be used for other RedHat Linux versions and Solaris. In Appendix A, installation notes specific to each operating system and platform are provided.

For software packages names, we use `<ver>` and `<arch>` tags instead of version numbers and architectures. For example, instead of

```
panc-3.0.2.i386.rpm
```

we will use

```
panc-<ver>.<arch>.rpm
```

For each package, install the latest version included in the quattor repository (see quattor web page) for the a given platform and quattor release.

2.3. BEFORE YOU START THE INSTALLATION!

This installation guide is still far from perfect! For easing your life, we **strongly** recommend you (before starting the Quattor installation) to have a look at the Quattor tutorial slides available under the following address:

quattor.org→documentation→tutorials→Quattor tutorial at CERN 1/2005

In particular the slide set called "Installing a Quattor Server and Client" provides useful information and notes, in particular for typical errors and problems you may run into when setting up Quattor. **Ideally you should print out the slides and have them next to you when following this installation guide!**. Note that the slides were prepared for release 1.0, whereas this installation guide describes release 1.1, but the differences are minimal.

2.4. INSTALLATION PROCEDURE

Quattor aims for a highly modular architecture, and it can be used for many different kind of environments, from very small sites with tens of nodes, to very large sites with thousands of nodes. Each site can have a customized installation of quattor adapted to its needs. In the Appendix B there are some examples of how to use quattor to manage sites of different sizes.

Quattor has been designed to coexist with legacy tools, services and procedures. Quattor aims to provide clean interfaces so it can be incrementally installed replacing the already existing installation and configuration utilities.

It is highly recommended to read this guide entirely before starting the installation of quattor.

The basic step to install a quattor server are:

- install a Configuration Database (Chapter 3.),
- install a Software Repository (Chapter 4.),
- install a basic Automated Installation Infrastructure server (Chapter 7.),
- load the default configuration templates (Chapter 5.),
- populate the Software Repository (Section 4.5.), and finally
- install the clients nodes (Section 7.2.).

As a general rule, you can assume that when a quattor software package installs a configuration file directly under `/etc`, this means that the default configuration values should work for most of the environments. However, if the package installs the configuration files under a documentation directory, this means that the **default values do not work**, and that you should edit and change the configuration parameters.

For a more advanced quattor environment, you may perform the following *optional* steps as well:

- create you own set of configuration templates (Chapter 5.),
- install the SQL server module (Chapter 8.), and
- install a Web Proxy for the software repository (Section 4.9.).

2.5. INSTALLATION USING APT AND YUM

For RedHat 7.3, RHES3 and Scientific Linux 3, it is possible to install Quattor using plain RPM's, but also using APT or YUM. This is documented in the Appendix D.

2.6. UPGRADING AN EXISTING QUATTOR ENVIRONMENT

If you already are running Quattor, you should read the appendix F for more information on upgrades.

2.7. QUATTOR FOR LCG-2 INSTALLATIONS

Note that this guide does not cover how to install LCG-2 with Quattor. However, you can find pointers and documentation on this subject in appendix E.

3. INSTALLING THE CONFIGURATION DATABASE

The *Configuration Database* (or CDB) is a central place to manage configuration information for client nodes. A CDB server is a machine running Scientific Linux, the quattor CDB server software, and a Web Server (for the sake of simplicity, in the rest of this document we will assume that we are using Apache as web server). Although not necessary, for security reasons it is highly recommended to install on the CDB server all the Scientific Linux package updates as well. For more information about CDB, please read the document [3], and CDB utilities man pages.

3.1. CDB INSTALLATION

Once you have completed the installation of the Scientific Linux distribution, and the Apache Web Server, you have to install the following quattor/CDB software packages:

```
perl-LC-<ver>.<arch>.rpm
perl-AppConfig-caf-<ver>.<arch>.rpm
perl-CAF-<ver>.<arch>.rpm
panc-<ver>.<arch>.rpm
cdb-<ver>.<arch>.rpm
```

Note: If you are using APT/YUM, please refer to Appendix D. The CDB packages are included by the `quattor-cdb` meta-package.

The `cdb` RPM creates a new user called the *Configuration Database Manager*. This user will be the owner of the database directory structure, and it will be used to manage CDB locally. By default the CDB manager user name is `cdb`, with group name `cdb`, and for security reasons, it is disabled. **Enable this account by changing its password**.

The configuration parameters for CDB software are stored in file `cdb.conf` under the directory `/etc`. Normally, the default parameters included in this file should be correct for most of the quattor installations. Edit the file, and change those parameters (if any) that do not fit to your environment. Comments explaining the meaning of each configuration parameter are provided in the file. The most important parameters are:

- `top`: that specifies the directory where the database will be stored, and
- `pan` and `cake`: that are the paths to the Pan language compiler and the Pan make utility.

The next step is to create and initialize the configuration database itself. In order to do that run, as user `root`, the CDB initialization script:

```
cdb-setup
```

Please, check that the initialization program has created a set of directories to store high (`hld`) and low level (`lld`) description templates, under the database directory (the directory you specified with the `top` configuration parameter, by default `/var/lib/cdb`). Note that the initialization script also creates a symbolic link from the Apache HTML directory (`/var/www/html/profiles`) to the `lld/xml` directory (`/var/lib/cdb/lld/xml`) to allow client nodes to download their configuration profile.

3.2. SYNCHRONIZATION WITH CDB

Some quattor modules (and perhaps other non-quattor or external programs as well) need to keep synchronized with the contents of CDB. That is, whenever there is a modification on the templates stored on CDB (new templates, removed templates or modifies templates) they want to get notified and see what happened. These notifications are managed by the *CDB Synchronization* subsystem. For more information about the CDB Synchronization subsystem refer to the document [4].

In case of this Installation Guide, we are interested in the synchronization subsystem because it is used by the Automated Installation Infrastructure (see Section 7.) and the SQL Server Module (see Section 8.) Later on, if you install these modules, you will have to change the CDB's list of machines to notify for synchronization.

The list of machines to get notified in case of changes on CDB is defined in the CDB's configuration file `cdb.conf`. For each machine to be notified we have to add an entry in the configuration file like:

```
server_module host.example.org
```

If there are no machines to notify, it has to be specified explicitly in the following way:

```
server_module none
```

3.3. MANAGING CDB

We use a remote management tool, called `cdbop`, from a different machine than the CDB server. The `cdbop` tool provides sessions management (commit or rollback changes), user authentication and authorization, shell-like editing, and so on.

It might happen that CDB contains old, unused information about sessions that have not been properly closed by CDB admins. To save space on disk, CDB provides a script, called `cdb-ses-clean`, that cleans up unclosed CDB sessions which are older than a configurable quantity of time (see the `cdb-ses-clean(8)` man page for more information). It is run by default every week via `cron`.

CDB Management

The management utility `cdbop` is based on a SOAP interface and allows remote access to the CDB. Install on the CDB server the following SOAP packages:

```
perl-SOAP-Lite-<ver>.<arch>.rpm
cdb-soap-server-libs-<ver>.<arch>.rpm
cdb-soap-server-<ver>.<arch>.rpm [2]
```

If you are using APT/YUM: These packages are included by the `quattor-cdb` meta-package.
Please restart your web server after having installed the above packages!
The list of users allowed to remotely manage CDB is stored in the file:

```
/etc/httpd/conf/.passwd
```

This file should contain one line per user allowed to connect to CDB, in the format:

```
login::encrypted_password
```

where `login` is the name of the user (note that `cdbop` users are different from the machine defined users) and `encrypted_password` is the encrypted password. Encrypted passwords can be generated with (literally):

```
openssl passwd
```

The utility `cdbop` uses an access control list (ACL) to provide permissions to users. The ACL file is stored on:

```
/etc/httpd/conf/cdb.allow
```

[2] Note that this package adds user `apache` to group `cdb`.

This file should contain one line per used allowed to connect to CDB, in the format:

```
user permission
```

Where `user` is the name of the user defined in the password file above, and the `permission` can be one of the following (See Section 5. for more information about the difference between *production* and *non-production* templates):

- empty: simple user;
- `rw`: the user is allowed to read and write non-production templates (those whose name does not start with "pro_");
- `admin`: the user is allowed to modify production templates.

Important note: On Apache/httpd, you need to edit the `httpd.conf` configuration file and to change the setting `'Timeout'` from 300 to 7200. This will avoid that your SOAP server is shut down after 300 seconds of execution.

Installing the CDB client software

Next step is to install the client software. On every machine that is going to be used remotely to manage CDB, install the following packages:

```
perl-LC-<ver>.<arch>.rpm
perl-AppConfig-caf-<ver>.<arch>.rpm
perl-CAF-<ver>.<arch>.rpm
perl-SOAP-Lite-<ver>.<arch>.rpm
perl-Text-Glob-<ver>.<arch>.rpm
perl-Term-Shell-<ver>.<arch>.rpm
perl-Term-ReadLine-Perl-<ver>.<arch>.rpm
perl-TermReadKey-<ver>.<arch>.rpm
perl-libwww-perl-<ver>.<arch>.rpm
cdb-cli-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-client` meta-package.

The configuration of the `cdbop` utility is managed in the global configuration file, which needs to be created:

```
/etc/cdbop.conf
```

This configuration file can be overwritten by a local, per user, configuration file located at user's home directory (mind that if you define a local configuration file, the global configuration file will not be read):

```
~/.cdbop.conf
```

You can find an example of a configuration file in the documentation directory:

```
/usr/share/doc/cdb-cli-<ver>/
```

Edit the example file, modify the configuration parameters according to your needs, and copy the file to its right place. Important configuration parameters you should modify, are the following:

- `server`: CDB server hostname, and

- `protocol`: set the protocol used by SOAP to perform the communication between server and client (by default it is `https`).

Then start the client running the command `cdbop`, provide a valid user and password, open a session with the `open` command, and list the available templates with the `list` command. For more information about `cdbop`, read its man page `cdbop(1)`, or type `help` on `cdbop` shell. Also a small usage tutorial is available in this guide (Appendix H).

CDB Recovery

For recovery purposes, it is possible to access directly the CDB with the `cdb-simple-cli` utility. You need to install on the CDB server node the package:

```
cdb-simple-cli-<ver>.<arch>.rpm
```

If you are using APT/YUM: This package is included by the `quattor-cdb` meta-package.

`cdb-simple-cli` should be used by the CDB manager, logging on the server machine as user `cdb`. A set of basic commands is available for adding, removing, listing, etc. templates. This utility is non-interactive, i.e., each command corresponds to an option given at the command line. For more information, read its man page entry.

4. INSTALLING A SOFTWARE REPOSITORY

The *Software Repository Server* (SWRep) allows system managers to create, and remotely control (add, remove, query, etc), a repository storing all software packages used to install clients. SWRep is useful also to generate HLDL templates with package lists (see Section 5.6.) For more information about SWRep, please refer to SWRep software man pages, and the document [5].

A Software Repository Server is a machine with the SWRep software installed, and with a directory containing all the software packages used for client installation. In the most simple configuration, the SWRep server and the CDB server are located on the same machine. For very large computing farms, the SWRep server can be replicated to other machines: in this cases, a load-balancing and proxy-caching architecture should be deployed as explained in Section 4.9.. The default protocol used by the clients to download software packages is HTTP, however, SWRep let us to use other alternative protocols, like FTP or NFS.

Important note: If you are planning to use other software distribution tools than SWRep/SPMA (like APT/YUM), you can skip this section (see also Appendix D on the differences between SPMA and APT/YUM).

4.1. PREREQUISITES

You need to have a web server running on the node hosting the SWRep repository. By default, we assume you will use apache ("httpd" RPM on Scientific Linux). Please make sure that this rpm is installed, and that the web server is running.

Important note: Due to a broken implementation in the RPM HTTP client, it is required to disable keepalives (The RPM client cannot handle HTTP/1.1 answers when the server uses keepalive). Failing to do so may lead to long SPMA/rpmt execution times.

You may need to add to the httpd server configuration:

```
<IfModule mod_setenvif.c>
     BrowserMatch "rpm/.*" nokeepalive force-response-1.0
</IfModule>
```

4.2. INSTALLATION OF SWREP SERVER SOFTWARE

Install the following quattor generic and SWRep software packages:

```
perl-LC-<ver>.<arch>.rpm
perl-AppConfig-caf-<ver>.<arch>.rpm
perl-CAF-<ver>.<arch>.rpm
swrep-libs-<ver>.<arch>.rpm
swrep-server-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-swrep` *meta-package.*

The `swrep-server` RPM creates a new user called the *Software Repository Manager*. The repository manager is a special Unix user that is the owner of the packages contained in the repository. The user is special in the sense you cannot login on to the repository server machine using this user, because it will have no login shell. By default the SWRep manager user name is `swrep`, with group name `swrep`, as home directory has `/var/swrep`, and as shell `/usr/sbin/swrep-server`. Besides to the repository manager, there must be a *Repository Location*, where software packages will be stored. It is recommended to use a repository located in a directory accessible from a web server, because clients will use by default the HTTP protocol to download the packages. The repository default location is:

```
/var/www/html/swrep
```

The configuration of SWRep is managed through the configuration file:

```
/etc/swrep/swrep-server.conf
```

You can find an example of configuration file in the documentation directory:

```
/usr/share/doc/swrep-server-<ver>/
```

Edit the example file, modify the configuration parameters according to your needs, and copy the file to its right place. Important parameters of `swrep-server.conf` file are:

- `name` and `owner` of the repository, used in configuration templates (see Section 5.),
- `url:` URL from where clients can download the software packages in the repository, you can use HTTP, NFS or FTP protocols for software package downloads, but HTTP is the recommended protocol,
- `aclfile:` Access Control List file (see Section 4.3. for more information about ACLs),
- `rootdir:` root directory of the repository.

The repository management is performed via the Secure Shell (SSH). You have to change the configuration of the SSH daemon on the SWRep server, thus edit the configuration file:

```
/etc/ssh/sshd_conf
```

and enable the option:

```
PermitUserEnvironment yes
```

and then restart the `sshd` server to make effective the changes with:

```
service sshd restart
```

4.3. MANAGING THE REPOSITORY

In those machines that are going to be used to manage the software repository install the following SWRep software packages:

```
perl-LC-<ver>.<arch>.rpm
perl-AppConfig-caf-<ver>.<arch>.rpm
perl-CAF-<ver>.<arch>.rpm
swrep-libs-<ver>.<arch>.rpm
swrep-client-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-swrep` meta-package.

The configuration of SWRep client is managed through the configuration file (which needs to be created):

```
/etc/swrep/swrep-client.conf
```

You can find an example of configuration file in the documentation directory:

```
/usr/share/doc/swrep-client-<ver>/
```

Edit the example file, modify the configuration parameters according to your needs, and copy the file to its right place. Important parameters of `swrep-client.conf` file are:

- `repository`: repository location in `user@host` format, where `user` is the `username` of the repository manager, for example we can use `swrep@host.example.org`,
- `ssh-params:` parameters used for the ssh communications, for example `'-2 -q -a -x -T -e none'`.

For each user allowed to manage the repository, we have to add, on the SWRep server, the user SSH public key (typically found in the file `~/.ssh/id_rsa.pub`) to the file:

```
/var/swrep/.ssh/authorized_keys
```

For example, if we want to use the user `user1` at host `host.example.org` to manage the repository, we should add something like the following to the `authorized_keys`:

```
environment="SSH_USER=user1" ssh-rsa AAAAB3NzaC1
YwP0jGlWvMl6D+LVIpCEU+qGhQdCNL691gvufcaK9jKwdfPB
VwlEbq8BpumdCOJlH3v8Q9i2hEqgCTT6WeRVxk+NCL5dQ6Ag
au1UpZ40= user1@host.example.org
```
[3]

Create your own key with the `ssh-keygen(1)` utility. *It is very important that you specify in the file the SSH_USER environment variable as well.*

The ACL rights to create, delete, and so on, packages is specified in SWRep server in the file:

```
/etc/swrep/swrep.acl
```

This file is managed though the `swrep-client` utility, but the first user, the one that should give rights to the remaining managers, should be added by hand. Edit the `swrep.acl` file, and add a new entry with the master user:

```
user1:/
```

When managing the software repository remotely, it is highly recommended to start an SSH agent [4] specially if you are going to do several operations. For example, with:

```
ssh-agent $SHELL
ssh-add
```

The utility to manage the software repository is `swrep-client`. You can perform a local management, in the same machine as the repository, or a remote management. If you want to perform a remote management, you should install the `swrep-client` and `swrep-libs` packages in the remote client machine. For more information read the man page `swrep-client(3)`.

[3] Note that the user `user1` can login from any machine, not only the `host.example.org` machine.

[4] Alternatively, you can use no *passphrase*, but this is a bit risky.

4.4. DEBUGGING THE SWREP CLIENT SETUP

You can check that your swrep client setup is correct by running `swrep-client listrights`. If you get prompted for a password, and not a passphrase, it probably means that you have your SSH connection not correctly setup. You should then run `swrep-client --verbose --debug 4 listrights` which shows you the actual SSH connection parameters. Probably you have some error in the `ssh-params` configuration option of swrep-client.

If you use ssh-2 keys, you may need to add an -2 to the ssh-params connection string. If this does not help, try to run on the server an ssh daemon in debugging mode on a different port (e.g. with `sshd -d -p 60000`) and then connect to it with the connection string from the client, adding -p 60000. This may help you understanding what is wrong with your setup.

Another typical mistake is to add line-breaks in the SSH public key added to the authorized_keys of the server. This should be avoided.

4.5. LOADING SOFTWARE PACKAGES

The repository is organized into *platforms*, and each platform into *areas*. All the packages belonging to a given platform should be for the same hardware architecture and operating system release. Given a platform, the packages can belong to any area, and it is up to the system manager how the packages are organized.

Note that the 'area' concept has no physical meaning (all packages belonging to a given platform end up in the same physical web server directory, independently of what 'area' they belong to). The 'area' concept is only used for ACL control (see above).

For example, we can create a new platform to store Scientific Linux 3 packages for the x86 architecture, with (only alphanumeric characters are allowed in platform's names):

```
swrep-client addplatform i386_sl3
```

And inside this platform, we can create the area `/base`, containing all those packages of the distribution CDs, and an area called `/updates`, to store the published packages updates. We can do that using the following commands:

```
swrep-client addarea i386_sl3 /base
swrep-client addarea i386_sl3 /updates
```

In the same way, we can create as many areas as needed. For example, we can create an area called `/extra` to store the software that is specific to our institute. But remember that the 'area' concept is just for setting ACL's (who is authorized to modify what area). All packages will appear in the same physical location, independently of the area they have been tagged to. It is perfectly possible to have only a single area and to tag all packages to belong to that area.

New packages are added to the repository with the `put` command, if packages are stored locally on our machine, or the `pull` command, if the package location is given by an URL. For example, to add package `foobar` to our repository you should use:

```
swrep-client put i386_sl3 /tmp/foobar-4.2-1.386.rpm /extra
```

4.6. BOOTSTRAPPING THE SWREP FOR THE FIRST TIME

Unfortunately, there is not yet an bulk update mechanism for adding multiple RPM's to a SWRep repository in one go via `swrep-client`.

An alternative is to use the `bootstrap` command of `swrep-client`. You need to copy over all your RPMs of your installation CD to the local directory **on the server** hosting the software repository (e.g. `/var/www/html/swrep/i386_sl3`). Then, you run for instance

```
swrep-client bootstrap i386_sl3 /base
```

which will register all the missing RPM's in the SWRep internal database, under platform `i386_sl3` and area `/base`. This is the easiest way of getting a large number of new RPM's into SWRep; the same command comes very handy for **updating** an area. (Please check the manpage of swrep-client for more information on this command.)

4.7. VERIFYING THAT ALL YOUR RPM'S ARE ON SWREP

You can verify that you have all your RPMs needed for your clients on the SWRep by using the `swrep-client list` command, for example: `swrep-client list i386_sl3`. Alternatively, you can point a web browser to the SWRep server: `http://swrepserver.mydomain/swrep` and verify that indeed your packages are all there!

Note also that of course, apart of the RPM's from the installation CDs and your own RPM's, and any update RPM's, you need to add all Quattor RPM's to the repository, as these RPM's are needed for installing clients!

4.8. REGULARLY UPDATING THE SWREP CONTENTS

Note that newly appeared RPM's need to be added to SWRep before they can be configured in CDB to be used by the SPMA software package manager client. Doing this manually can be a boring exercise, in particular taking into account the large number RPMs released containing fixes for security holes...!

There is a small shell script shipped with the SWRep client RPM called `swrep-mirror` which can help you mirroring a directory containing RPM's to SWRep (read the file `README.swrep-mirror`).

4.9. USING PROXY SERVERS WITH SWREP

If your farm is composed of *less* than $\tilde{8}$00 nodes you can skip this Section.

In order to provide reliable, redundant and load balanced access to software repositories, a proxy-caching system can be adopted in either a *forward* or a *reverse* configuration. With the former mode, a client asks the proxy server to fetch for him an object located on the origin server; this configuration is non-transparent since the client must be properly configured. In the latter mode, a client talks to the proxy server as if it were the origin server, which again will fetch the requested object from it: this configuration is opaque as only a different repository name must be known to the client. Forward proxies are commonly used to provide geographical access and caching to users residing in the same area, whereas reverse proxies are more suited to cater (external) users wishing to access an intranet environment, and also it is simple to implement cluster load-balancing and/or name-space partitioning with them.

Quattor client and server software is by design reverse proxy compatible. For example, SPMA supports the usage of both forward and reverse proxy servers. Given the reasons above, the natural choice for a proxy-caching system in a quattorized farm is the reverse configuration.

Proxy servers can be used to build SWRep and AII (see Section 7. for more information about AII) server hierarchies. It is possible to have a central set of (load-balanced) servers, and then per rack, to have a 'head-node' which acts as proxy server for software downloads (see Figure 3). The goal of this two-tier caching architecture is to reduce the load on the backend servers when an object, such as an RPM file, is requested several times: indeed, this may happen during a cluster-wide installation/update, when all the nodes involved will download from the repository the same set of packages. A two-tier reverse proxy-caching system works, roughly speaking, as follows:

- client nodes are configured to download objects from a head-node rather than from the frontend server(s);
- similarly, each head-node will download objects from the frontend server(s) rather than from the backend server (the software repository);

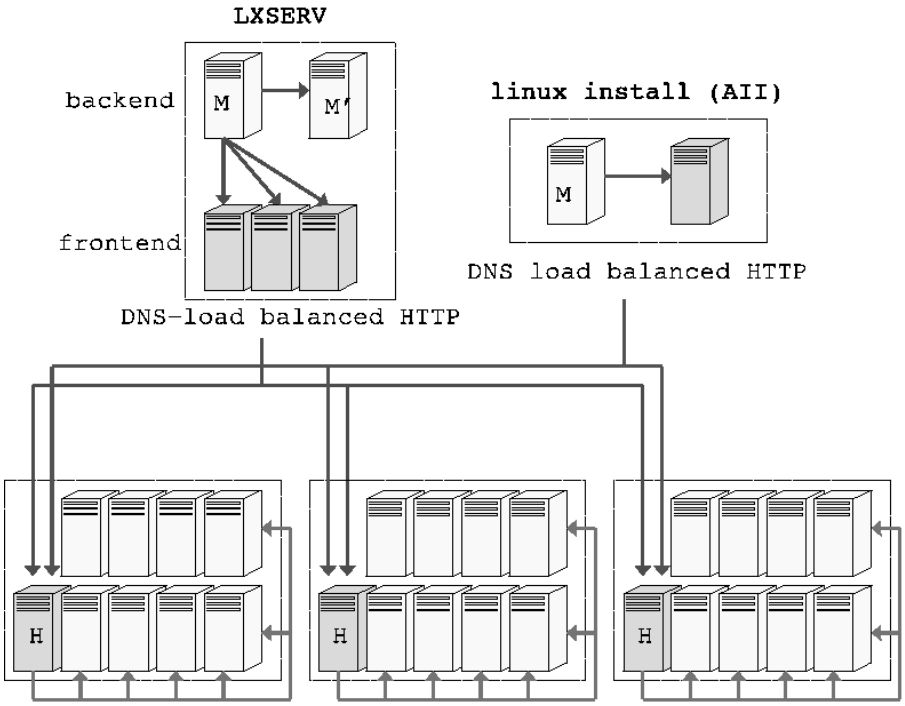

Figure 3: Proxy based Deployment

- each frontend server downloads objects from the backend server;
- both head-nodes and frontends have a cache for objects passing across them. This means that at the very first request for an object from a client, data will be downloaded from the backend server, and cached along the way on both a frontend and a head-node: at each further reference of the same object, the request will be "short-circuited" on the nearest cache holding that object, thus avoiding replicated transfers from the backend.

Proxy-caching systems can be implemented with several software products. In this guide, details about the configuration and the deployment of an Apache-based reverse proxy is provided in Appendix J. Alternatively, the Squid Web Proxy Cache could be used: this solution will be evaluated soon in a Quattor-managed cluster.

4.10. ALTERNATIVES TO SWREP

You do not have to install the SWRep software to have a working Software Repository for quattor. The only thing you need is a collection of software packages accesible through a web server. However, we strongly recommend to use SWRep, because the management of the repository is greatly simplified.

If you choose not to install the SWRep software, perhaps you will find useful some auxiliary programs provided by the quattor software utility package `quattor-sw-util-<ver>.noarch.rpm`:

- `html2pan`: This command retrieve the html file from the given URL and parses any embedded anchors for references to RPMs. The script produces a valid repository PAN template for use with quattor.
- `html2updates`: This command retrieve the html file from the given URL and parses any embedded anchors for references to RPMs. The script produces a valid PAN template for use with quattor which contains `pkg_ronly` calls for each rpm. This is appropriate for an "updates" template for an OS.

- `rpmcheck`: This script takes an LLD file generated by the CDB component of quattor, extracts an rpm list by transforming the XML, and runs a fake installation of those rpms to ensure that the list works correctly.
- `comps2pan.xls`: This software is intended to transform a `comps.xml` file (used to describe package groups for RedHat-like systems) into a set of PAN templates suitable for use with quattor.
- `rpmq2pan_pkg`: This script queries the local RPM database for all the installed packages and writes a PAN template holding the corresponding package list.

Note: If you decide to use SPMA without SWRep but using a different web server setup you still need to ensure that keepalives are disabled as described in section 4.1..

5. CREATING A CONFIGURATION PROFILE

In Chapter 3. it was explained how to setup and run the CDB server software. The next step is to load into CDB those templates (written in Pan language) that compose the configuration profiles of our client nodes.

This chapter [5] describes a set of default Pan configuration templates. These templates should be adequate for most of the small and medium size environments. However, if you are planning to install and manage a large network with quattor, it is recommended to define a new set of templates adapted to your needs. You can use the templates described here as a starting point to create your own customized templates.

5.1. THE PAN LANGUAGE

For setting up your configuration, it is convenient to have at least a basic understanding of the PAN language. A reference guide for PAN, as well as a tutorial, can be found on the Quattor web site. We recommend you to have a look at it!

5.2. LCG TEMPLATES

NOTE: If you plan to install LCG software, you can find a complete template set at the LCG Quattor Working Group home page. See Appendix E for more details.

5.3. DEFAULT TEMPLATES

All the templates mentioned in this section (except component templates) can be found in the RPM file

```
pan_templates-<ver>-noarch.rpm
```

This RPM, as well as each NCM component, installs the templates under the directory:

```
/usr/share/doc/pan_templates
```

The `standard` subdirectory contains all those templates that are common for most of the installations. System managers probably do not need to change these templates, just load them. The set of templates under the `site_specific` directory contains all those templates which the system manager definitely should change, since they do contain very specific information about local environments. They are provided just as an example, and can be used as an starting point to create a new set of customized templates. The `components` subdirectory contains those configuration templates related to configuration components. Note that the templates for configuration components are not shipped in the pan-templates RPM but in each component's individual RPM. All these templates will be explained in the remainder of this section.

As it was explained in Section 3., in order to load a template into CDB you can use the remote management utility `cdbop`. For example, the following procedure adds a template called `my_template.tpl`:

```
$ cdbop
quattor CDB CLI: Version 1.8.14

Enter user-name: <username>
Enter password: <password>

Connecting to http://mycdbserv...
```

[5] Many thanks to Martin Bock for the help with this chapter.

```
Welcome to CDB Command Line Interface
Opening session...
Type 'help' for more info

[cdb] ~ > add my_template.tpl
[INFO] adding template: my_template
[cdb] ~ > commit
[cdb] ~ > exit
$
```

Before adding a new template to the template's database, CDB performs a set of basic checks. For example, CDB checks if the Pan syntax is correct, if all the other referenced templates are available, and so on. Due to these checks, you have to load all the templates needed to describe the configuration of a node before creating the profile of that node.

It is recommended to load the templates into CDB in the order that are described in this section.

For a broader overview of `cdbop` usage see Appendix H.

5.4. STANDARD TEMPLATES

In this section we are going to describe very briefly the set of proposed *standard templates*. Note that there is no need to change these!!

Functions for data types and structures validation: These templates define a set of utility functions for the validation of the data types and structures (described in the following sections).

```
pro_declaration_structure_validation_functions
```

Common data types: These templates declare the most common data types used in the rest of the configuration templates, like for example, types for dates and IP addresses. Data types are atomic elements, and cannot be composed of other data types.

```
pro_declaration_types
```

Common data structures: These templates declare the most common data structures used in the rest of the configuration templates, like for example, structures for CPU and NIC description, a structure for describing software packages, and so on. Structures are composed out of other types (like the ones defined above, or pan internal types) and other structures.

```
pro_declaration_structures
```

Units and other functions: These templates define some other useful functions, and some units to be able to enter the configuration information in a more human readable format.

```
pro_declaration_units
pro_declaration_functions_filesystem
pro_declaration_functions_network
pro_declaration_functions_general
```

Declaration of Profile's Base: This template declares the base of all the configuration profiles.

```
pro_declaration_profile_base
```

5.5. TEMPLATES FOR HARDWARE DESCRIPTION

The `pro_hardware_*` templates are used to describe the hardware of our client nodes. On the directory

```
/usr/share/doc/pan_templates/site_specific
```

you can find an example of a common hardware configuration. You can copy and then modify these templates to describe your own hardware. Note that hardware descriptions are more useful if you install the SQL Server Module, eg. for running inventories (see Section 8.).

```
pro_hardware_card_nic_intel_e100
pro_hardware_ram_1024
pro_hardware_harddisk_STD_80
pro_hardware_cpu_GenuineIntel_Pentium_4_2600
pro_hardware_asus_terminator_p4_533a
            -> final node including above definitions
```

Note: You can skip hardware definitions as they are optional, except for the hard disk definition. The hard disk definition is required for checking partitioning information. See inside the `pro_hardware_asus_terminator_p4_533a` template for how to do this.

5.6. TEMPLATES FOR SOFTWARE MANAGEMENT

Note: You can skip this section (5.6.) if you do not plan to use SPMA/SWRep for software management!

The software management templates have been designed in such a way that CDB can perform some validation tests, for example, that software packages have valid names, or that all the packages mentioned in the profiles are available in a software repository. These checks are performed with the aid of the types definitions and validation functions contained in the standard set of templates (described in Section 5.4.). In Figure 4 you can see the relationship between all the templates that deal with software management.

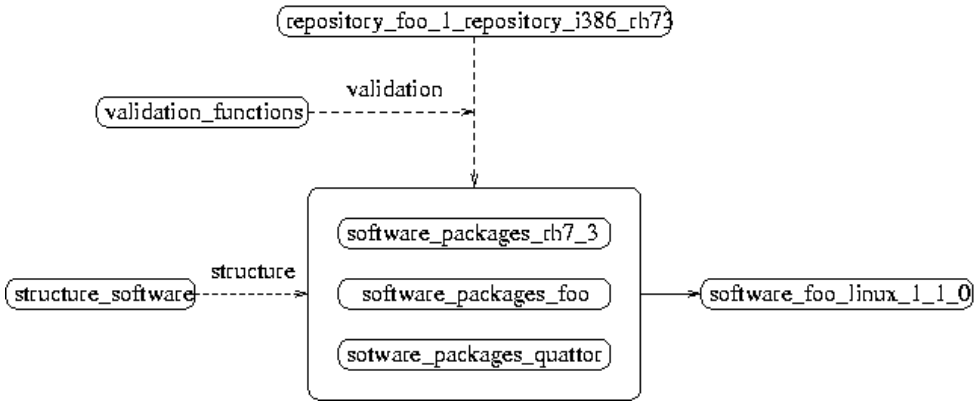

Figure 4: Templates for Software Management

The first thing we have to do is to load the template describing our software repository and the packages it contains. This template **must be** created using the `template` command of the SWRep client utility which you have installed previously. Use:

```
swrep-client template i386_sl3 > repository_myrep_i386_sl3.tpl
```

for generating the template corresponding to the architecture ‘i386_sl3’ on the repository ‘myrep’.

Important note: Due to a bug, the output may contain extra heading lines coming from system login messages. Please comment the "Banner" option, if it is enabled, in the `sshd` configuration file (`/etc/ssh/sshd_config`) on the SWRep node. Also check for and delete any other line before the starting comments in the outputted template! Failing to do so will result in an invalid template.

It is mandatory that the name of the repository (`myrep`) and the name of the architecture (`i386_sl3`) are in the template name as shown above.

The generated template contains a list of all packages which are on the SWRep for the specified platform. This information is required for checking the validity of the pkg_add,repl statements. For more information about SWRep read Section 4. and the `swrep-client(1)` man page. An example of repository description can be found in the template:

```
repository_myrep_i386_sl3
```

For an easier software template management, it is recommended to organize the software packages into groups. A simple organization can have, for example, three groups: standard Scientific Linux 3 packages, quattor packages, and those extra packages needed by local users ("user"). For each group we must have a template describing its packages:

```
pro_software_packages_i386_sl3
pro_software_packages_quattor_i386_sl3
pro_software_packages_user (if required)
```

Of course you can have as many groups==templates as you wish and require.

Finally, we should have a template to put together all the information needed to install software packages on the clients:

```
pro_software_mycluster
```

This template also references the repository template discussed above, thus, remember to use the correct name.

PAN functions for package list manipulation

Some PAN functions are provided for manipulating the package list in the profile, which are used in the above-mentioned templates:

- `pkg_add("packagename","version-release","arch");`
  Adds a package to the profile.
- `pkg_del("packagename",["version-release"]);`
  Removes a package version (or all if none specified) from the profile.
- `pkg_repl("packagename","new ver-rel","arch"[,"old ver-rel"]);`
  Replaces the package version 'old' by 'new' in the profile. If 'old' is not specified, it replaces all other versions.

It is important to understand that the `pkg_add`, `pkg_del` and `pkg_repl` functions do only modify the final list of desired packages. Eg. `pkg_del` does not instruct to delete any package, but removes it from the list of desired packages Useful when modifying inherited profiles. More information about these functions can be found in the template: `pro_declarations_functions_general.tpl`

Note that it is possible to simultaneously install multiple versions of a package (like multiple kernels, or multiple glibc versions for i686 and ia64, etc). This can be done the following way:

```
"/software/packages" = pkg_add("glibc","2.3.2-95.33","i686");
"/software/packages" = pkg_add("glibc","2.3.2-95.33","ia64","1");
```

The "1" indicates that multiple versions of that package should be allowed. Otherwise, an error is reported.

If you are planning to create your own software templates, what we recommend to do is to install a target machine with all the desired software, and then create a template describing this machine with the aid of the query facilities of the `rpm` utility, for example with the command (on a single line!):

```
rpm -qa --queryformat '["/software/packages" =
    pkg_add("%{NAME}","%{VERSION}-%{RELEASE}","%{ARCH}");\n]'
```

You could put the lines given as output in your `pro_software_packages_<arch>`, for example. Note that you may need to add the "multiple versions" parameter as described above, in case you have multiple versions of packages installed on your reference node! A better way to generate package templates from a local RPM database is to use the program `rpmq2pan_pkg` coming with the quattor software utility package `quattor-sw-util-<ver>.noarch.rpm`. This tool can also handle multiple package versions, either automatically or interactively.

Default package versions

Having multiple clusters, each with similar but not identical software setups makes maintaining the list of `pkg_add/pkg_repl/pkg_del` statements very difficult, in particular when security updates need to be applied and version numbers need to be replaced across many templates.

In order to ease this task, it is possible to define separately which are the default package versions for a given platform. This is done in the following templates:

`pro_software_packages_defaults_redhat_7_3` (RH7.3)
`pro_software_packages_defaults_sl3` (SL3)

This way, it is sufficient to include that template and to use `pkg_add` without having to specify the version number, e.g.:

```
pkg_add("httpd");
# instead of
# pkg_add("httpd","2.0.46-44.ent","i386");
```

In order to keep your system secured, is of course **important to keep the version information defined in the default templates up-to-date!!**

5.7. CONFIGURATION COMPONENT TEMPLATES

For each service managed by quattor, you have to load into CDB the corresponding configuration template containing the configuration parameters for this service. In the documentation of the configuration components (read the man pages of each component) it is described which templates are necessary to load to use the component. The templates related to configuration components are stored in the directory:

```
/usr/share/doc/pan_templates/components
```

Note: the component templates are *not* contained in the pan-templates RPM, rather in each individual component's RPM.

In this section we are going to see an example of this process, loading the templates necessary to use the `grub(8)` configuration component and the SPMA's configuration component.

You need the following base templates for the management of configuration component (these templates have been described in Section 5.4. and they should be already loaded):

```
pro_declaration_structure_validation_functions
pro_declaration_structures
pro_declaration_types
```

Then, load the template describing the grub and SPMA configuration components:

```
pro_declaration_component_grub
pro_declaration_component_spma
```

And then, the templates with the default configuration parameters:

```
pro_software_component_grub
pro_software_component_spma
```

Normally, you do not need to modify these templates to change the configuration parameters of a component, because this configuration information can be overwritten and managed in a per-site, per-cluster or per-node basis (see Section 5.8. for more information).

Since the component templates are shipped with the component RPM's (`ncm-<component>`), it is recommended to install the required component RPMs on the node from which you run cdbop. (Note that installing the RPM of a component does not activate the component.)

5.8. TEMPLATES FOR SITES AND CLUSTERS CONFIGURATION

By default, there are three different places where you can change the configuration information. Depending on which level you use, the changes will have a wider or narrowed effect.

The first level is the site configuration template. If you modify a configuration parameter in this template, all the nodes managed by quattor in this site will be affected. The meaning of *site* is something that the system manager has to decide, for example, it could be a computer room, a faculty or an entire campus. Site changes are made in the template:

```
pro_system_base
```

The second level is the per-cluster (a group of client nodes) configuration template. You should have one of this templates for each cluster type managed by quattor. Assuming a cluster named "mycluster":

```
pro_system_mycluster
```

There is a example template called `pro_system_mycluster` which you should carefully inspect.

The third level is the configuration profile of the node itself. This level will be described in the next section.

Note that if required, you can add more levels, for defining sub-clusters or sub-sites for example. You can in fact have as many levels as you need!

5.9. CREATING A NEW CONFIGURATION PROFILE

In this section we are going to see how to create the configuration profile of a new client node. **This is pre-requisite to any client configuration or installation with Quattor!**

The first step is to create a new Pan "object" template called `profile_myhost` where `myhost` is the name of the client node you want to install. You can use as an example the `profile_mynode` template shipped with the default template set.

Edit the new template and modify it according with the configuration information of the new node. For a very simple setup, you need to modify:

- The file name of the template (`profile_myhost.tpl`).

- The name of the template inside the file (`object template profile_myhost;`).
- The ethernet hardware address (`/hardware/cards/nic/eth0/hwaddr`).
- The IP address (`/system/network/interfaces/eth0/ip`).
- The host name (`/system/network/hostname`).

Of course, many other templates and parameters can (and should) be changed for a complete and coherent setup, but setting just the above parameters allows to check if the CDB installation, and the initial set of templates added to CDB, are OK.

Load the new template into CDB ("add" command, then "commit").

If everything goes well, CDB should generate the LLD profile (based on XML) of the new client node. Otherwise the CDB CLI will generate an error message.

You can check that the profile has been created by looking at directory:

```
/var/www/html/profiles
```

You should be able to access the profile also via HTTP:

```
http://mycdbserver/profiles/profile_myhost.xml
```

Once this step is completed, the client node can be installed, and then configured.

5.10. IMPORTANT ADVICE

If you plan to use SPMA for software management (see section 5.6.), please keep the list of packages (default package versions) updated according to the latest security releases!

Tracking down common errors

If you do not want to use SPMA/SWRep, do not forget to comment out the line including the software definitions ("`include pro_software_mycluster`") inside the `pro_system_mycluster` template.

If you use SPMA/SWRep, the packages defined in the templates will be validated against the repository contents. If one or more packages selected with pkg_add or pkg_repl is/are not found in the repository, an error similar to the following will be produced when you have produced a profile and you run commit on cdbop:

```
cannot find any repository holding the following packages
name: quota version: 3.10-4 arch: i386
name: kernel version: 2.4.21-20.0.1.EL arch: i686
name: kernel version: 2.4.21-27.0.2.EL arch: i686
name: ddd version: 3.3.1-23 arch: i386
name: ncm-spma version: 1.2.17-1 arch: noarch
name: grub version: 0.93-4.3 arch: i386
name: ncm-grub version: 1.2.12-1 arch: noarch
...
```

You need to ensure that the missing packages have been uploaded to the Software Repository, and that the repository template has been regenerated using swrep-client template, and updated in CDB.

Another error which may hit you is if you try to include twice a package in a profile with pkg_add. If you want to replace a package already included in the profile with a different version, you *must* need pkg_repl. Otherwise, the following error may be generated when you run CDB commit:

```
*** user error:  pkg_add: package emacs-lisp found twice in profile,
    but no multiversion specified
```

6. MANUAL INSTALLATION OF CLIENT NODES

In this section it is explained how to install the quattor client software on a client node. Once the software has been installed and started, quattor will take the control of the configuration of the node. Note that the procedure described in this section can be fully automated by using the Automated Installation Infrastructure (AII) described in Section 7.

If you plan to install quattor on many nodes, it is highly recommended to use AII instead of this manual installation procedure!

This section is intended for those system managers that do not want to use AII (for example because they want to use their own automatic installation system).

It is however recommended to carefully read this section, in order to get a deeper understanding of how the Quattor client works.

6.1. INSTALLING THE QUATTOR CLIENT SOFTWARE

In this section we are going to see which quattor software packages we have to install into client nodes. For each package, we will explain very briefly the purpose of the software, and we will provide references for further information. Note that in this section we are going to describe how to perform a minimal installation of quattor. Once quattor is running, the remaining client packages will be installed by quattor itself.

The quattor modules related to the management of client nodes are the *Configuration Cache Manager* (CCM), that caches locally the profiles describing the configuration parameters of the node; the *Node Configuration Manager* (NCM), responsible to deploy the desired configuration on the node; and the *Software Package Management* (SPM) for software management (installations, removals and upgrades).

The first step is to install in the client node the following list of external software packages, and quattor generic libraries:

```
perl-Compress-Zlib-<ver>.<arch>.rpm
perl-LC-<ver>.<arch>.rpm
perl-AppConfig-caf-<ver>.<arch>.rpm
perl-CAF-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-client` meta-package.

Configuration Cache Manager

The *Configuration Cache Manager* (CCM) caches locally the profiles describing the configuration parameters of the node. CCM also includes the `ccm-fetch` application, that retrieves XML profiles from the CDB server, the `ccm-purge` application, that cleans up the configuration cache directory from unused configuration profiles, and the NVA-API library, which provides methods to query the configuration parameters contained in the profiles. In Figure 5 there is a scheme of how these programs relate to each other (note that the `listend` daemon will be explained in Section 6.1.4.). For more information about CCM please read [6], and `ccm-fetch(8)` and `ccm-purge(8)` man pages. The NVA-API library is described in [7] and [8].

Install the following CCM package:

```
ccm-<ver>.<arch>.rpm
```

If you are using APT/YUM: This package is included by the `quattor-client` meta-package.

The configuration parameters that control the behavior of CCM are stored in the file:

```
/etc/ccm.conf
```

You can find an example of configuration file in the documentation directory:

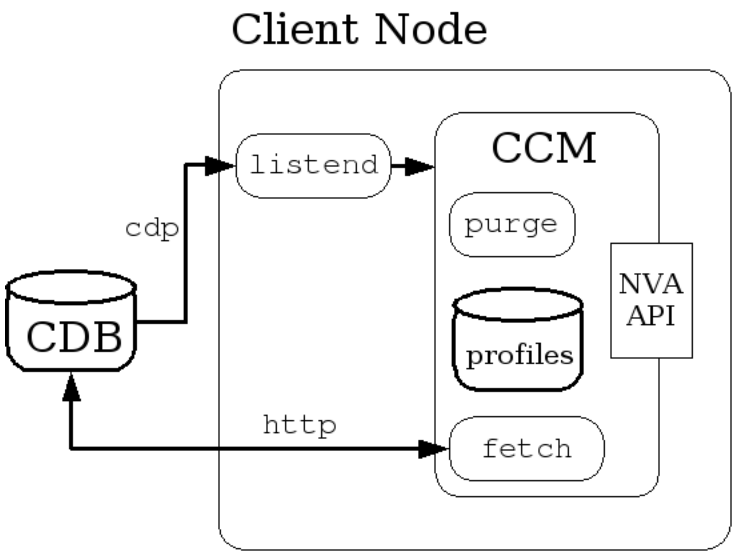

Figure 5: Configuration Cache Manager

```
/usr/share/doc/ccm-<ver>/eg
```

Edit the example file, modify the configuration parameters according to your needs, and move the file to its right place. Important configuration parameters you should take special care, are the following:

- `profile`: specify the URL from where the client can download its XML configuration profile. It is composed of the CDB's URL, plus the name of the profile.
- `cache_root`: the local cache directory where the profiles will be stored.

Note that in this configuration file you can use the variables `$host` and `$domain`, that will be replaced for the `hostname` and `domainame` values of the client node. For example, the entry:

```
profile http://cdb.mydomain.org/profiles/profile_$host.xml
```

on a machine called `myhost` will be interpreted as:

```
profile http://cdb.mydomain.org/profiles/profile_myhost.xml
```

The final step is to run the CCM initialization script:

```
ccm-initialise
```

Node Configuration Deployer

The *Node Configuration Deployer* (ncd) is the framework and front-end for executing the *configuration components*. ncd is part of the *Node Configuration Manager* (NCM) subsystem. In Figure 6 there is a scheme of how the elements that compose NCM relate to each other (note that the `cdispd` daemon will be explained in Section 6.1.5.). For more information about ncd please read [9], and the `ncm-ncd(1)` man page. In this section we are going to see how to install and configure ncd. It is also recommended to install the tool `ncm-query(1)`. This tool is useful to solve problems, by querying the configuration parameters of the profile stored locally in the node.

Install the following NCM packages:

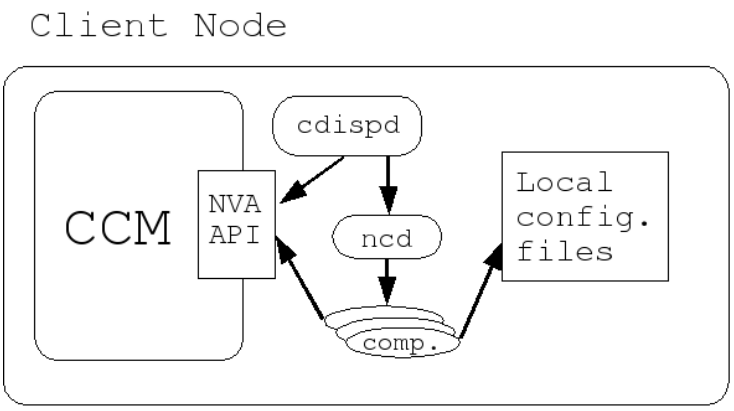

Figure 6: Node Configuration Manager

```
ncm-ncd-<ver>.<arch>.rpm
ncm-query-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-client` *meta-package.*
The configuration parameters that control the behavior of ncd are stored in the file:

```
/etc/ncm-ncd.conf
```

The default configuration parameters should work for most of the environments. Edit the file, modify the configuration parameters according to your needs. The most important configuration parameter is `cache_root` that should point to the root of the cache directory of CCM if it is not in its standard location (see Section 6.1.1.).

Software Package Management Agent

The *Software Package Management Agent* (SPMA) is responsible for the installation, upgrading and removal of software packages. It uses `rpmt`, a transactional version of RedHat package manager `rpm`. In Figure 7 there is a scheme of how quattor manages the installation of software packages on clients. For more information about SPMA please read [10], or the man pages `spma(1)` and `rpmt(8)`.

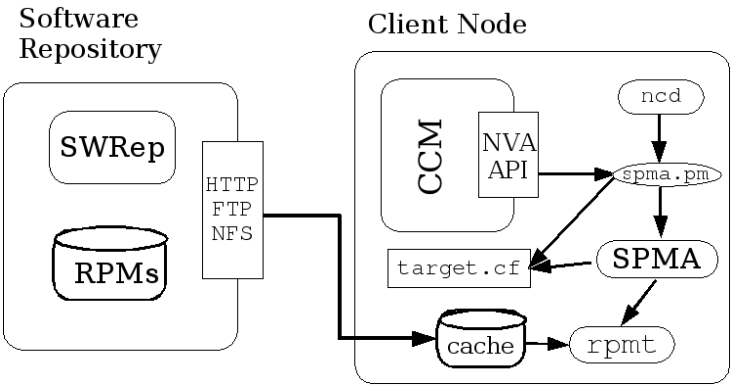

Figure 7: Sotware Packages Management

Note that SPMA is an optional package. If you need so, you can use other tools for the software management in client nodes (like for example apt-get or YUM). See Appendix D for more information on the differences between APT/YUM and SPMA. If you are not going to use SPMA, you can skip the SPMA instructions below.

Install the following SPMA packages:

```
rpmt-<ver>.<arch>.rpm
spma-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-spma` meta-package.
Note that it is not recommended to run SPMA and APT/YUM at the same time.

Important note: It is known that `rpm` versions 4.2.2 (SL302) and 4.2.3 (SL303) have a bug in the HTTP transport. If you experience errors like "BAD RPM key" when downloading a package with 'rpm' via HTTP, you can either downgrade the RPM packager to version `4.2.1` or upgrade it to the version `4.2.3-21_nonptl` or higher (this is recommended). Please, use the corresponding RPMT package `rpmt-2.0.4-1.<arch>.rpm`. Both these packages can be found in the Quattor software repository.

The configuration parameters that control the behavior of SPMA are stored in the file:

```
/etc/spma.conf
```

The default configuration parameters should work for most of the environments. Edit the file, modify the configuration parameters according to your needs. Important configuration parameters you should take special care, are the following:

- `userpkgs`: allows to install packages in nodes without quattor.
- `localcache`: improve package downloading and installation synchronization, by caching locally the packages before installation.

Finally, install the NCM configuration component responsible of SPMA configuration:

```
ncm-spma-<ver>.<arch>.rpm
```

If you are using APT/YUM: This package is included by the `quattor-client` meta-package.

Configuration Distribution Protocol Daemon

The `listend` daemon implements the Configuration Distribution Protocol (CDP). Upon receiving a CDP notification it runs the `ccm-fetch` or `cdbsync-nch` applications. For more information about CDP and `listend` please read the `listend(8)` man page.

Install the following CDP `listend` package:

```
cdp-listend-<ver>.<arch>.rpm
```

If you are using APT/YUM: This package is included by the `quattor-client` meta-package.
The configuration parameters that control the behavior of `listend` are stored in the file:

```
/etc/cdp-listend.conf
```

The default configuration parameters should work for most of the environments. Edit the file, modify the configuration parameters according to your needs. Important configuration parameters you should take special care, are the following:

- `port`: listening port for the daemon.
- `nch`: (optional) executable location for `cdbsync-nch`.
- `fetch`: executable location for `ccm-fetch`.

Configuration Dispatch Daemon

The Configuration Dispatch Daemon (`cdispd`) is part of the *Node Configuration Manager* subsystem (see Figure 6). `cdispd` waits for new incoming configuration profiles. In case of changes with respect to the previous profile, `cdispd` will invoke the affected components via the `ncd` program. For more information about `cdispd` daemon, please read [9], or the `cdispd(8)` man page.

Install the following `cdispd` package:

```
ncm-cdispd-<ver>.<arch>.rpm
```

If you are using APT/YUM: This packages is included by the `quattor-client` meta-package.

The configuration parameters that control the behavior of `cdispd` are stored in the file:

```
/etc/ncm-cdispd.conf
```

The default configuration parameters should work for most of the environments. Edit the file, and modify the configuration parameters according to your needs.

6.2. RUNNING QUATTOR FIRST TIME

Once you have created the LLD configuration profile for the new client (see Section 5.), and you have installed all the quattor client packages described above, you can start quattor. The first time, we have to launch quattor manually, but since then, all the changes will be performed automatically, without system manager intervention.

Download the configuration profile of the client to the local cache:

```
ccm-fetch
```

And then, start the Configuration Dispatch Daemon:

```
service ncm-cdispd start
```

After a while, the machine should start to reconfigure itself to match the configuration described by its profile.

Please refer to Chapter 9. for more information how to actually deal with Quattor in operation.

7. AUTOMATED INSTALLATION OF CLIENT NODES

The *Automated Installation Infrastructure* (AII) is a set of tools to install quickly and automatically client nodes using standard services like DHCP, TFTP and KickStart. For more information about AII, please read the document [11], and the AII utilities man pages, `aii-shellfe(8)` and `aii-installfe(8)`.

AII manages three services:

- DHCP: to distribute network configuration parameters,
- TFTP: to distribute Network Boot Program (NBP, the boot loader), and kernel images, and
- OS installer tools: to manage operating system installation.

Currently, AII supports ISC-DHCP as DHCP sever, PXElinux as NBP, and Anaconda and KickStart for the installation of RedHat/Scientific Linux.

Please, note that AII *is not functional on IA64 and has not yet been tested on X86_64*. Contributors are welcome!

7.1. INSTALLING AN AII SERVER

AII has been designed to support redundancy / load reduction by spreading installation services in different computers. However, in case of small environments, you can concentrate all the services in just one machine. The most simple environment typically has a machine running all quattor services (CDB + SWRep + AII), or two servers, one for quattor CDB and SWRep, and the other one for AII.

This section describes how to install all the AII services (DHCP + TFTP + OS Installer) on the same server machine. This machine can be the machine where you have already installed CDB and SWRep, or a different one. In Section 7.3. we will see how to install an advanced AII server by replicating AII services among different machines.

AII Software Installation

AII needs the following quattor generic packages and libraries (if you are installing AII in the same server machine than CDB or SWRep, they should be already installed; if you are using a different machine, you have to install them):

```
perl-LC-<ver>.<arch>.rpm
perl-AppConfig-caf-<ver>.<arch>.rpm
perl-CAF-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-aii` meta-package.

AII also uses CCM (see Section 6.1.1.) to download and query the profiles of all the client machines to install. Install the following packages:

```
perl-Compress-Zlib-<ver>.<arch>.rpm
ccm-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-aii` meta-package.

Note that in the particular case of AII, it is not necessary to initialize or configure CCM, because this will be automatically done by the AII tools. Once you have installed CCM, the last step is to install the AII server software package:

```
perl-XML-Simple-<ver>.<arch>.rpm
cdbsync-<ver>.<arch>.rpm
ncm-template-<ver>.<arch>.rpm
aii-server-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-aii` meta-package.

DHCP Server Configuration

AII needs an ISC DHCP server already installed and configured on the installation server machine. For more information about how to install and configure ISC DHCP, please check the web page [12], or read the man page `dhcpd.conf(5)`. Note that the corresponding RPM is named 'dhcp' in Scientific Linux 3.

AII has been designed to preserve the already existing configuration information for the DHCP server. That is, AII will add, remove or modify only the information of those hosts managed by AII, preserving the configuration information of those hosts not being managed.

If you plan to configure the DHCP server only with AII, it may be useful to use the DHCP configuration file example that comes with the AII package. This example can be found at:

```
/usr/share/doc/aii-<ver>/eg/dhcpd.conf
```

Edit this example file, insert the configuration parameters about your network (read the comments on the file), and copy the file to the `/etc` directory.

Finally, you should review the configuration file `/etc/aii-dhcp.conf`. However, unless you are using a non-standard ISC DHCP installation, probably the default configuration values for `aii-dhcp` are suitable.

PXELinux Configuration

The Network Boot Program used by AII to bootstrap a Linux kernel into the nodes through the network is *PXELinux*. PXELinux is part of the SysLinux project (see [13] for more information). Please, check that you have already installed the `syslinux` version 2 or higher, if not, install or upgrade the SysLinux package.

AII uses the TFTP [6] protocol to download the kernel loader (PXELinux) through the network. You should have the `tftp-server` and the `xinetd` packages installed. By default, the directory that TFTP will use to download the kernel loader is `/osinstall/nbp`.

You don't have to modify the configuration of PXELinux, because this is managed by AII automatically. The PXELinux configuration file template used by AII is located at:

```
/usr/lib/aii/nbp/i386_rh73_pxe.conf (RH7.3)
/usr/lib/aii/nbp/i386_sl3_pxe.conf (SL3)
```

This default PXELinux configuration file template should be adequate for most of the AII server installations. Copy, rename and modify this template if you are planning to use a non-standard AII server (it is possible to define via CDB which configuration file template to use, this is explained later on).

The NCM Template library is used by the AII to generate the PXE and KickStart configuration files. For more information on this Quattor library a man page is available by typing `man NCM::Template`.

Finally, we have to create a new directory, called `/osinstall/nbp/i386_sl3` (`/osinstall/nbp/i386_rh73` for RH7.3) where to put those files that Scientific Linux needs to bootstrap the operating system on a new node. The files to copy are `vmlinuz` and `initrd.img` (and also `initrd-everything.img` if you are installing RH7.3). You can find them on any Scientific Linux/RedHat installation CD-ROM (for SL3, it is the disk #3). For example, you can copy those files with:

```
cp /mnt/cdrom/images/pxeboot/* /osinstall/nbp/i386_sl3
```

[6] Please mind that the `tftpd` daemon will query the hosts database for access control information, so it may happen that you need to change the `/etc/hosts.allow` and `/etc/hosts.deny` files. See the `hosts_access(5)` and `tftpd(8)` man pages for more information.

KickStart Configuration

AII uses KickStart to perform an initial installation of the operating system on clients, and to install the quattor client software. You don't have to create the KickStart configuration files, because this is managed by AII automatically. The configuration file template used by AII to generate the KickStart files is located at:

```
/usr/lib/aii/osinstall/i386_sl3_ks.conf
```

If required, you can copy this template KS file for generating your own template KS files. The NCM Template library is used by the AII to generate the PXE and KickStart configuration files. For more information on this Quattor library a man page is available by typing `man NCM::Template`.

The resulting KickStart configuration files will be stored at `/osinstall/ks`. You have to make this directory accessible through your HTTP server, for example with:

```
ln -s /osinstall/ks /var/www/html/ks
```

Adding the Linux distribution CD and exporting it via HTTP

In order to perform remote installs, the contents of the Scientific Linux installation CD-ROM must be copied onto the AII server and made accessible using the HTTP protocol. You need to first create a directory:

```
mkdir -p /var/www/html/i386_sl3
```

This directory will contain the CDROM contents. You must copy the whole contents of the CD-ROM with:

```
cp -a /mnt/cdrom/* /var/www/html/i386_sl3
```

Note: If you want to save space *and if your AII server is the same than the SWRep server*, you can reuse the packages of your already populated SWRep repository, by first deleting the RPMS directory copied from the CDROM, and then creating a symbolic link to the SWRep RPM directory:

```
rm -rf /var/www/html/i386_sl3/SL/RPMS

ln -s /var/www/html/swrep/i386_sl3/ /var/www/html/i386_sl3/SL/RPMS
```

It goes without saying that you need to do this for every platform you want to manage (SL3 x86_64, RH73, etc)!

Acknowledgments Server

When a client node finish to install itself, it sends a notification message to the AII server. In this way, the AII server can change the configuration of the node to boot from the local disk the next time it reboots. If the client fails to send the notification message, it will get re-installed during the next reboot.

Copy the acknowledgment cgi-bin script to your Apache web server cgi-bin directory:

```
cp /usr/sbin/aii-installack.cgi /var/www/cgi-bin
```

and configure the `sudo` utility in such a way that `apache` user can run AII commands without providing passwords. For example, add the following entry to the `/etc/sudoers` file:

```
apache your_hostname=(ALL) NOPASSWD: /usr/sbin/aii-shellfe
```
[7]

[7] this reads: the `apache` user on the host `your_hostname` can run `aii-shellfe` with the privileges of ALL users without being asked for his or any other password.

7.2. AII MANAGEMENT

The configuration of the DHCP server is managed with the `aii-dhcp(8)` tool; the configuration of PXElinux with `aii-nbp(8)`; and the configuration of KickStart files with `aii-osinstall(8)`. Although the `aii-dhcp`, `aii-nbp`, and `aii-osinstall` tools can be used as stand-alone programs, they are normally used through the `aii-shellfe` front end (see Section 7.2.1.), or the `aii-installfe` remote management front end (see Section 7.2.2.). For each AII tool, there is a configuration file that describes its behavior. For example, for the `aii-dhcp` tool, there is the configuration file `aii-dhcp.conf`. If you use a standard AII installation, probably you do not need to modify those AII tools configuration files.

The configuration parameters that control the behavior of the local AII management tool are stored in the file:

```
/etc/aii-shellfe.conf
```

You can find an example of configuration file in the documentation directory:

```
/usr/share/doc/aii-<ver>/eg
```

Edit the example file, modify the configuration parameters according to your needs, and move the file to its right place. A very important configuration parameter you should take special care is the `cdburl` parameter, that should point to the URL where XML profiles are available.

AII uses a set of configuration files "*templates*" in order to create the real configuration files for those services needed for its operation. These template files come with a set of configurable tags (enclosed by $< >$) that will be replaced with the configuration parameters stored in the profile of each node managed by AII. These configuration parameters are stored in the pan templates

```
pro_software_component_aii.tpl (general)
pro_software_component_aii_i386_rh73.tpl (RH73 specific)
pro_software_component_aii_i386_sl3.tpl (SL3 specific)
```

These Pan templates can be found in `/usr/share/doc/pan-templates/components`. Examine them carefully, and copy them into your CDB using cdbop and change those configuration parameters according to your needs. In particular, you should fill in the `server` and `cdb` parameters of the `osinstall` structure with the hostname of your OS installation server, and CDB server. You may also need to change the 'nbp/template' and 'osinstall/template' entries if you decide to use other templates for configuring PXE or KickStart. (See Section 5.7. for more information about how to change configuration parameters in CDB.)

AII Local Management

The AII server can be managed locally with the help of the `aii-shellfe(8)` tool. If you want to perform a local management, you must be connected into the AII server, with the `root` user.

In order to add a new node to be managed by AII, we have to use the `--configure` option of the `aii-shellfe` tool. Use for updating the configuration of a node ("configuring" a node) as known to AII:

```
aii-shellfe --configure node_name
```

Note: The node has to be defined in DNS! Otherwise aii-shellfe will fail.

Note: Everytime the configuration of a node is updated in CDB, you need to run `aii-shellfe --configure`, unless you enable the automatic synchronization with CDB which is described later.

Note: AII expects the network interface used for initial boot to be configured in the CDB profile to have the 'boot' flag set to 'true' (for example, `"/hardware/cards/nic/eth0/boot" = true`). If no interface has this flag set, AII will fail.

Once the new node has been configured, we can manage the node via AII. For example, we can mark the node for its complete installation the next time it reboots with:

```
aii-shellfe --install node_name
```

In the same way, we mark a node to boot from local disk (no installation) during next reboot with:

```
aii-shellfe --boot node_name
```

Or we can query the status (marked to boot from local disk, or for installation) of a node with:

```
aii-shellfe --status node_name
```

If we want to manage more than one node in just one single step, AII support PERL like regular expressions instead of node names. For example, To prepare configuration files for grid001, grid002, grid005 use:

```
aii-shellfe --configure grid00[125]
```

For more information about how to manage AII locally, please read the man page `aii-shellfe(8)`.

AII Remote Management

The AII servers can be managed remotely with the help of the `aii-installfe(8)` tool. This tool accepts the same options than the `aii-shellfe(8)` local management tool. That is, it can be used to configure, boot, install or query the status of any host managed by an AII server. The `aii-installfe` tool runs remotely, via SSH and `sudo`, on any remote AII server, or list of AII servers.

Install the following AII client package on the machine being used for the remote management:

```
aii-client-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-client` meta-package.

Then, on each remote AII server to be managed, you have to chose an user and change its SSH configuration to allow to run `aii-shellfe` remotely, and the configuration of the `sudo` utility to run `aii-shellfe` with `root` permissions. How to configure SSH has been explained on Section 7.1.6., and how to configure `sudo` in Section 7.2.2.

With the `--servers` option you can specify the AII server to be updated or queried. Use the `@` symbol to select the remote user. If there are more than one server, use a blank space ' ' to separate them:

```
aii-installfe --servers user1@server1 user2@server2 ...
```

You can use the `aii-installfe` tool to configure a local AII server as well. Just use `localhost` as hostname:

```
aii-installfe --servers localhost ...
```

In this case, the `aii-shellfe` tool will be executed directly on the local machine (no SSH required). If you want, you can mix local and non-local servers:

```
aii-installfe --servers localhost user1@server1 ...
```

For more information about `aii-shellfe` please refer to its man page `aii-shellfe(8)`.

Automatic Synchronization with CDB

AII can be configured to keep itself in synchronization with the contents of CDB. That is, whenever there is a change in the profiles of CDB, AII will get a notification, and it will adapt itself to the changes: If there are new profiles, AII will authomatically configure the new profiles, and will mark them for installation, without the intervantion of the system manager. In the same way, if the configuration profile of a node has changed, AII will reconfigure that node. And if a profile has been removed, AII will remove the configuration of the node.

As it was explained in Section 3.2., the first step to keep AII in sync with CDB is to modify the configuration of CDB and add a new *server module*. Edit the `cdb.conf` file and add a new line:

```
server_module aii_server
```

Where `aii_server` is the hostname of our AII server. Then, on the AII server we have to install the CDB Synchronization module and the Configuration Distribution Procotol daemon (described on 6.1.4.. Install the following packages:

```
cdbsync-<ver>.<arch>.rpm
cdp-listend-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-aii` meta-package.
The configuration parameters that control the behavior of CDB Sync are stored in the file:

```
/etc/cdb-sync.conf
```

You can find an example of configuration file in the documentation directory:

```
/usr/share/doc/cdb-sync-<ver>/eg
```

Edit the example file, modify the configuration parameters according to your needs, and move the file to its right place. Important configuration parameters you should take special care, are the following:

- `url`: URL of CDB profiles list
- `actions`: list of local action programs

In case of AII synchronization, you should add the action:

```
action = aii-shellfe --notify
```

7.3. SETTING UP AN ADVANCED AII SERVER

In this section we are going to describe how to set up an advanced AII server, that may be usefful in case of large sites.

Installing Different OS

You can use just one AII server to install more than one version of the operating system. For each OS version, you need:

- the OS installer
- the list of RPMs needed for the initial installation
- the configuration templates (e.g. `i386_sl3_ks.conf` and `i386_sl3_pxe.conf`), and
- modify accordingly the template `pro_software_component_aii_i386_sl3.tpl`.

Using More than one AII Server

AII has been designed to support redundancy / load reduction by spreading installation services in different computers. A possible scenario is the following:

- 1 server for CDB
- 1 server for DHCP + NBP (TFTP server) + acknowlegements server
- 1 server for OS installation

This possibility is just an example, e.g. you can have more than one OS installation server, or you can split the TFTP and DHCP server. If you change the layout you should install and configure the AII software on each server, and update accordingly the nodes profile (template `pro_software_component_aii_i386_sl3.tpl`).

Note however, that scalability problems are highly unlikely with AII, unless you decide to reinstall your 600+ node farm at once! Scalability issues are more likely to appear on the SWRep side where regular software updates need to be propagated in a short time window on all nodes.

8. SQL SERVER MODULE INSTALLATION

The purpose of the *SQL Server Module* program is to convert all the profiles available at a specified CDB server to SQL relational form. Every time the SQL Server Module runs it updates the relational database appropriately to reflect the latest set of profiles stored on the CDB. For more information about the SQL server module read the `cdb2sql(8)` man page.

Note: There is no dependency on the SQL server module by other Quattor services. The SQL server module is *not* required for running the Quattor CDB, SWRep, CCM/NCM nor SPMA.

8.1. SQL SERVER INSTALLATION

The SQL Server is a machine with a relational database (RDBMS) for which the Perl DBI framework driver (see `http://dbi.perl.org` for more information) has been installed. The default DBI driver is `mysql`, for the MySQL system (note that the SQL server module has also been tested with the Oracle driver on a Oracle 8i installation).

You must also ensure that some database (called for example `cdb`) exists in the chosen RDBMS, with privileges SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE and DROP on `cdb.*` granted to a distinguished user (which, by default, is the user invoking `cdb2sql` program).

For example, in case of MySQL you should do something like [8]:

```
create database cdb;
grant select, insert, update, delete, index, alter, create,
drop on cdb.* to USER@localhost;
```

Where USER is replaced by the relevant user name.

SQL Server Module needs the following generic libraries and quattor packages. If you are installing SQL Server Module in the same server machine than CDB, some of them should be already installed. If you are using a different machine, you have to install all of them:

```
perl-Compress-Zlib-<ver>.<arch>.rpm
perl-DBI-<ver>.<arch>.rpm
perl-HTML-Parser-<ver>.<arch>.rpm
perl-MIME-Base64-<ver>.<arch>.rpm
perl-TimeDate-<ver>.<arch>.rpm
perl-XML-Parser-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-cdbsql` meta-package.

Also, you have to install the dababase driver. In case of MySQL the driver is:

```
perl-DBD-MySQL-<ver>.<arch>.rpm
```

CDB Synchronization Notification System is used to keep in sync with the contents of CDB (see 3.2. for more information about how to install the synchronization subsystem). Install the following package:

```
cdb-sync-<ver>.<arch>.rpm
```

And add the following action to the `cdbsync.conf` configuration file:

```
actions = cdb2sql.wrapper
```

Once you have installed the generic Perl libraries and CDB-Sync, the last step is to install the SQL Server module software package:

[8] Depending on your RDBMS it's possible that these SQL commands may work as well.

```
cdb2sql-<ver>.<arch>.rpm
```

If you are using APT/YUM: These packages are included by the `quattor-cdbsql` *meta-package.*

The `edb-cdb2sql` requires a directory in which it can cache XML profiles (compressed) between invocations. By default this is:

```
/var/lib/cdb2sql/cache
```

which the `cdb2sql` process must have permission either to create or to write to. An alternative directory can be supplied with `--cachedir` option.

The configuration parameters that control the behavior of SQL Server Module are stored in the wrapper script:

```
/usr/bin/cdb2sql.wrapper
```

Edit the script and modify the configuration parameters according to your needs. Important configuration parameters you should take special care, are the following:

- `PW_FILE`: specify the password file and additional options

8.2. SQL SERVER MANAGEMENT

The password and settings file (by default `/etc/cdb2sql.settings`) should contain definitions for the database user (variable `CDB2SQL_USER`) and password (variable `CDB2SQL_PASSWD`), for example:

```
CDB2SQL_USER=cdb2sqluser@cdb
CDB2SQL_PASSWD=mysecretpw123
```

If there are any other options needed for your database, provide them in the variable `CDB2SQL_OPTIONS`, for example in case of MySQL you could use:

```
CDB2SQL_OPTIONS="--database=cdb --debug"
```

and then, turn the file to root read-only (remember, it contains the site specific settings, including the database password).

```
chmod 600 /etc/cdb2sql.settings
```

9. QUATTOR IN OPERATION

Up to now, we have seen how the Quattor modules are installed, and configured. However, there is not much information in this installation guide on how to actually use Quattor in operation. We will work on this (and your feedback is appreciated!). For the time being, apart of this installation guide, there is a tutorial mentioned at the beginning of this document which contains more relevant operational information:

quattor.org→documentation→tutorials→Quattor tutorial at CERN 1/2005

There are four slide sets which may particularly be useful:

- "Installing a Quattor Server and Client"
- "Management of PAN configuration templates and profiles"
- "Software Management with Quattor"
- "Introduction to NCM Configuration components"

You will find information on how to use SPMA and NCM, look up relevant log files, write NCM components, and relevant exercises to test out.

10. SCDB

This section explains the usage of 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. This alternative[9] , 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.

10.1. 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 accustomed 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.

[9] 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" simultaneously. Because of this need and a pre-existing subversion infrastructure at the laboratory, LAL developed and uses this alternative configuration database.

10.2. 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[10].

apache The subversion server runs within an Apache2 server. The code is available from the Apache Foundation[11] 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[12] 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[13].

eclipse Eclipse is optional, but provides a very good front-end for graphical management of a site's configuration. See the eclipse web site[14] 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.

10.3. 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.

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.

IMPORTANT: when using deploy and post-commit scripts, make sure that ant in the temporary directory is executable.

The post-commit hook script does the following:

[10] http://subversion.tigris.org/
[11] http://www.apache.org/
[12] http://java.sun.com/
[13] http://ant.apache.org
[14] http://eclipse.org/

1. Creates or updates a workspace on the subversion server with the revision just tagged.
2. Does a complete build of the configuration profiles.
3. For successful builds, it copies the generated profiles to 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.

10.4. SERVER CONFIGURATION

Subversion provides an excellent book[15] which describes both the use and administration of a subversion server. Although a standalone subversion repository can be accessed locally and a working SCDB can be constructed using local file access to the repository, the most common access method for subversion is web-based.

Assuming you have an Apache2 webserver installed, the next step is to install subversion and its apache module (called `mod_dav_svn` in SL). Check the subversion website for installation instructions for your distribution.[16]

Additionaly, the J2SE Development Kit (JDK) 5 (try http://java.sun.com/j2se/1.5.0/download.jsp) and `panc` are needed.

Configuring the webserver

The following section is taken from the subversion installation instructions and from chapter 6 of the subversion book. (Depending on your distribution or installation method, you might have an example configuration file in `/etc/httpd/conf.d/subversion.conf`.)

First, your httpd.conf needs to load the `mod_dav_svn` module. Verify that this line exists in your httpd.conf:

```
LoadModule dav_svn_module modules/mod_dav_svn.so
```

(otherwise apache will give an error like `"Unknown DAV provider: svn"`).
NOTE: if you built `mod_dav` as a dynamic module as well, make sure the above line appears after the one that loads `mod_dav.so`.

Next, add this to the *bottom* of your httpd.conf:

```
<Location /svn/repos>
DAV svn
SVNPath /absolute/path/to/repository
</Location>
```

In case of multiple repositories, one can use the `SVNParentPath` option.

This will give anyone unrestricted access to the repository through the link

```
http://your.webserver.something/svn/repos.
```

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

[16] RH/SL/CentOS-users can get their packages from `http://rpmforge.net/`. Install the correct rpm from `http://rpmforge.net/user/packages/rpmforge-release/`, do `apt-get update` and then `apt-get install svn mod_dav_svn`.

mod_svn provides webbased browse capabilities through xml style sheets. In the http/svn DocumentRoot create a directory svn and copy the files svnindex.css and svnindex.xsl in there. The path can be changed in the configuration with SVNIndexXSLT.

To actually create a repository (eg in /var/www/svn), go to the directory and run the following commands[17]:

```
svnadmin create stuff
chown -R apache.apache stuff
```

This will give you the repository stuff with SVNPath /var/www/svn/stuff.

Refining security to SCDB

A first step to better security is to use some basic authentication provided by the mod_auth module, by adding the following to the Location block:

```
AuthType Basic
AuthName "Subversion repository"
AuthUserFile /my/svn/user/passwd/file Require valid-user
```

The AuthUserFile directive sets the name of a textual file containing the list of users and passwords for user authentication. Don't put the file in the Documentroot as it contains the hashes of the passwords. Each line of the user file contains a username followed by a colon, followed by the crypt()-encrypted password. For more info on mod_auth see http://httpd.apache.org/docs/1.3/mod/mod_auth.html.

Even better security can be achieved with https-only access to your repository. Https is enabled through mod_ssl. A configuration example can be found in /etc/httpd/conf.d/ssl.conf. Don't forget to enable the SSL checking by issuing the SSLVerifyClient require statement. Also pay attention in ssl.conf to use the correct virtual host, set the servername to the servers FQDN, enable HostnameLoopups and set the document root to your svn repository. Make sure that this repository is not accessible through the Documentroot of other configurations (eg that from the default httpd.conf).

A good alternative to creating users and passwords is to use certificate based authentication. You can get certificates for your server and your users from a common trusted CA, or you can setup such a CA yourself (see eg Appendix I for this). To add certificate support, configure ssl.conf by setting SSLCertificateKeyFile, SSLCertificateFile, SSLCACertificatePath (don't forget to create the symlinked hashes (eg look in in /etc/http/conf/ssl.crt/Makefile).), SSLCARevocationPath (files in here must be updated as much as possible, also don't forget to run make -f Makefile.crl in that directory). Also make sure that the user who runs httpd has access permissions to all the certificate-related files.

Data inside the certificate can be used to make additional checks and/or to enable access control. For this, the information inside the the certificate must be "exported" to the other modules. The basic ways are FakeBasicAuth, ExportCertData and StdEnvVars. See the documentation of mod_ssl for more information. When using certifiacte based authentication, also set the SSL certifiacte options and use SSLVerifyClient require.

tip: set SSLVerifyClient none OUTSIDE the Location tags in case of strange error svn: PROPFIND of '/svn': Could not read status line: SSL error: sslv3 alert unexpected message.

Access control can be implemented using Limit statements

[17] Note that these are also the commands needed for a local file-based svn repositry.

1. For a read/write restricted repository: `Require valid-user`
2. For a write restricted repository:

```
 <LimitExcept GET PROPFIND OPTIONS REPORT>
Require valid-user
</LimitExcept>
```

3. For separate restricted read and write access:

```
AuthGroupFile /my/svn/group/file

            <LimitExcept GET PROPFIND OPTIONS REPORT>
                Require group svn_committers
            </LimitExcept>

            <Limit GET PROPFIND OPTIONS REPORT>
                Require group svn_committers
                Require group svn_readers
            </Limit>
```

Finer access control is allowed using `mod_authz_svn`. Make sure it gets loaded in `httpd.conf`:

```
LoadModule authz_svn_module modules/mod_authz_svn.so
```

You can use it by adding `AuthzSVNAccessFile /path/to/access/file` to the location. The syntax of the access file is the same familiar one used by `svnserve.conf` and the runtime configuration files. Lines that start with a # are ignored. In its simplest form, each section names a repository and path within it, and the authenticated usernames are the option names within each section. The value of each option describes the user's level of access to the repository path: either `r` (read-only) or `rw` (read-write). If the user is not mentioned at all, no access is allowed.
To be more specific: the value of the section-names are either of the form `[repos-name:path]` or the form `[path]`. If you're using the `SVNParentPath` directive, then it's important to specify the repository names in your sections.

Combination of `mod_auth_svn` with certificates can be a bit of a mess, because one has to pass a user to `mod_authz_svn`. From httpd version `2.0.51+` there's a directive `SSLUserName` that can be used for that, for all lower versions, one must use `FakeBasicAuth` together with the basic authentication module. For every user one must create an entry in the `AuthUserFile` with password `xxj31ZMTZzkVA` (it is the hash of the word `password`) and as user the full certificate DN[18].
Because this DN contains the `=`-sign, users can't be used directly in the `AuthzSVNAccessFile`. One must add them to groups and give all pernissions through group statements.

A working example of `mod_authz_svn` using certificates is then:

ssl.conf:

```
<Location /repos>
    DAV svn
    SVNParentPath /var/www/svn
    SVNIndexXSLT "/svn/svnindex.xsl"
    AuthzSVNAccessFile /etc/httpd/conf/authz
```

[18] Get eg the DN with `openssl x509 -noout -subject -in cert.pem`.

```
        SSLVerifyClient require
        SSLOptions           +FakeBasicAuth
        SSLRequireSSL

        AuthType Basic
        AuthName "test server"
        AuthUserFile /etc/httpd/conf/users
        Require valid-user

    </Location>
```

/etc/httpd/conf/authz:

```
[groups]
## Spaces are allowed, commas are the only separator
group1 = /C=XX/O=YY, /O=ZZ/OU=AA
[repos1:/]
*=r
@group1=rw
```

/etc/httpd/conf/users:

```
/C=XX/O=YY:xxj31ZMTZzkVA
/O=ZZ/OU=AA:xxj31ZMTZzkVA
```

Initial SCDB structure

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[19]

```
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"
```

[19] This is using the commandline svn client. For the people using local file based access, use `file:///path/to/where/you/issued/svn/create`.

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`.

10.5. 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.

NOTE2: make sure that when checked out, ant is executable!

Basic

The basic client setup consists of a commandline svn client, panc and your favorite editor. The first step is to do a svn checkuot of the repository. When setup correctly, this will provide you with a structure containing everything needed. To configure svn with certificate authentication, make sure you have the following structure in your `~/.subversion/servers` files:

```
[groups]
somegroup = some.full.host, *.some.domain

[somegroup]
ssl-client-cert-file = /bla/bla/bla.p12

[global]
ssl-authority-files = /bla/bla/bla/ca_local.crt
```

When connecting to the repository for the first time, svn wil prompt for the password of your key (It will also ask if you want the client to remeber it so you don't have to type it everytime).

Eclipse

Although commandline tools can provide you with everything you need, a graphical frontend can help overview the full structure of your pan-tree. Eclipse can be used as such a tool. (Although this is only tested using linux, eclipse is also supported for windows and mac. Please send some feedback when you

succeed in setting it up for these OSes.)
Fetch eclipse from `http://www.eclipse.org/downloads/index.php` (and make sure you have the java jdk_1.5.0 and panc). If you have both 1.4.2 and 1.5.0, make sure to make eclipse use 1.5.0. Things to check:

- Set JAVA_HOME to the 1.5.0 directory
- Make sure that the path to java points to the 1.5.0 version (eg `export PATH=$JAVA_HOME/bin:$PATH`).
- In eclipse, use `Window->Preferences->Java->Installed JREs` to make sure 1.5.0 is used internally.

To make eclipse work with SCDB, some additional software for eclipse needs to be installed. This can be done easily using the build in `Software updates`. Really needed is `Subclipse` that can use commandline svn as svn-client, but recommended is the use of `JavaSVN`. Note that it still is handy to have a commandline svn client installed.
Two more packages can added to have some nicer user features. These are `Sunshade` for parsing panc output and `Colorer` for pan syntax-highlighting.

In eclipse, go to `Help->Software Updates->Find and Install`. Select `Search new features to install` and continue with `Next->New Remote Site`. There you need to add the following (you can add them all at the same time):

Subclipse url: `http://subclipse.tigris.org/update`

JavaSVN url: `http://tmate.org/svn/`

Sunshade url: `http://sunshade.sourceforge.net/update`

Colorer url: `http://colorer.sf.net/eclipsecolorer/`

Then finish with some combination of `Finish->Select all software needed->Next->Accept->Next->finish->Install all->Restart eclipse`.
Some postconfiguration is needed to make everything work:

JavaSVN go to `Window->Preferences->Team->SVN` and select JavaSVN as SVN interface.
When using JavaSVN in combination with certifiactes, you must add the password in the `~/.subversion/servers` file[20]. Make sure that this file has correct access rights (eg `chmod 400`). A full example looks like

```
[groups]
somegroup = some.full.host, *.some.domain

[somegroup]
ssl-client-cert-file = /bla/bla/bla.p12
ssl-client-cert-password=hahahahaha

[global]
ssl-authority-files = /bla/bla/bla/ca_local.crt
```

Sunshade go to `Window->Preferences->Sunshade->Errorlink` and add the following reg-exp rules:

[20] This is probably fixed in latest version.

```
\[pan-syntax\]\s*(.*\.tpl):(\d+):
\[pan-compile\]\s*(.*\.tpl):(\d+)
\[pan-syntax\].*[\s\(](.*\.tpl)
```

Colorer As Colorer has no standard support for pan templates, this must be added. Get the `pan.hrc` from Quattor CVS (in `elfms/quattor/utils/eclipse`) and copy it in `/path/to/eclipse/plugins/net.sf.colorer_0.7.1/colorer/hrc/auto`. Then add to `eclipse/plugins/net.sf.colorer_0.7.1/colorer/catalog.xml` the following line:

```
<location link="hrc/auto/pan.hrc"/>
```

Now also add an association between `.tpl` and colorer: go to `Window->Preferences->General->Editors->File Associations`, add `.tpl` and associate the `colorer editor` with it. Now restart eclipse.

The only steps that rest are selecting the SVN repository, checking it out and setting up ant.

- `Window->Show view->Other->SVN->SVN repository` and add the new repository.
- browse the repository and go to the root of `deploy/tags/trunk` structure. Checkout the whole structure as a project (use rightclick)
- one has to add the `.project` file to `svn ignore`: `rightclick->Team->Add to svn:ignore`[21]
- in navigator, browse to `build.xml` file, check if all directories are set ok. Then go to `Window->Show View->Ant` and add the `build.xml` from the trunk. All ant tasks are now waiting to be used.
- start editing ;)

10.6. 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.

[21] Depending on the view depth, `.project` might not show up. Make sure that you can browse these so-called hidden files with navigator.

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.

10.7. 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.

10.8. TYPICAL TASKS

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

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.

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.

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.

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.

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.

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.

A INSTALLATION NOTES

This section deals with the installation and configuration of quattor on platforms other than Scientific Linux 3 (SL3). *Please read the `ReleaseNotes` file that comes with the quattor distribution to see the latest version of this section.*

Please mind that quattor has been designed to be able to install and configure multiple client platforms using just one single quattor server. In order to do that, quattor uses a set of naming conventions to distiguish which plaform we are installing. If you use quattor to install a different platform than SL3 on i386, some changes might be necessary. For instance, for RedHat 7.3 (RH7.3), you should create a software repository platform with the command:

```
swrep-client addplatform i386_rh73
```

Also, there are some files and directories whose names you may need to change:

- PXELinux cofiguration file base:
  `/usr/lib/aii/nbp/i386_rh73_pxe.conf`
- KickStart configuration file base:
  `/usr/lib/aii/osinstall/i386_rh73_ks.conf`
- OS installation bootstrap programs:
  `/osinstall/nbp/i386_rh73`
- OS installation base:
  `/var/www/html/i386_rh73`

A1. REDHAT LINUX 7.3

Here follows a list of changes with respect to Scientific Linux 3.

Pan Templates for Software Packages

Use the following set of pan templates:

```
pro_software_packages_redhat_7_3
pro_software_packages_quattor
```

And modify the contents of the following template to include the previous ones:

```
pro_software_mycluster
```

AII and DHCP Server configuration

Mind that RH7.3 comes with ISC DHCP version 2 whose configuration options may differ with respect to ISC DHCP version 3 (SL3). Please, read the comments in the `dhcdp.conf` sample configuration file distributed with quattor to see what options you should include depending on the DHCP server version used.

B EXAMPLES OF QUATTOR INSTALLATIONS

In this section we provide some hints about the installation of quattor for diferent site sizes.

B1. SMALL SITES INSTALLATIONS

For up to some 500-800 nodes, we recommend to use:

- A server machine with CDB + SWRep + AII

B2. MEDIUM AND LARGE SITES INSTALLATIONS

For a fabric composed of more than 800-1000, we recommend to use:

- A server machine with CDB
- A server machine with SWRep
- One or more SWRep replicas with Web cache proxies
- A server machine with AII
- A server machine with the SQL Server Module

B3. SITES RUNNING LCG-2 SERVICES

This guide does not cover how to install LCG-2 with Quattor. You can find documentation on this subject in appendix E.

Component	Description
ncm-accounts	NCM component to manage the accounts on the machine
ncm-cron	cron scripts management component
ncm-grub	configures the GRUB boot-loader
ncm-interactivelimits	configures system limits for interactive users
ncm-ntpd	ntpd configuration component
ncm-spma	generates the target SPMA file /var/lib/spma-target.cf

Table 2: Core Configuration Components

C CONFIGURATION COMPONENTS

This Section deals with the Configuration Components available in quattor. To the installation purpose, we distinguish two kinds of components: *Core Configuration Components* and *Optional Configuration Components*. The core components are those installed by default with quattor, because they are needed by most installations; conversely, optional components are not bundled with quattor and should be installed indivdually by hand.

C1. CORE CONFIGURATION COMPONENTS

The core configuration components are listed in Table 2. They are included in the template

```
pro_system_base
```

with a set of default configuration parameters. You can override this configuration parameters in several places, depending on if you want to do per-site changes, per-cluster changes or per-node changes (see Section 5.7. for more information).

ncm-grub

This component configures the `grub` (GRand Unified Bootloader) loader. It allows to select different kernels to be loaded by default. For more information see the `ncm-grub(8)` and the `grub(8)` man pages.

ncm-spma

This component configures the Software Package Management Agent (SPMA), the module that installs, removes or updates the packages in the client node. For more information see the `ncm-spma(8)` man page and Section 6.1.3..

ncm-interactivelimits

This component configures system limits for interactive users and groups. It edits the file:

```
/etc/security/limits.conf
```

This file is read by Pam module `/lib/security/pam_limits.so` and the values defined there are respected. See `pam(8)` and `ncm-interactivelimits(8)` for more information.

ncm-accounts

This component manages the accounts and groups in the client machine. It creates, modifies or removes users and groups as specified in the profile. For more information read the `useradd(8)`, `userdel(8)` and `ncm-accounts(8)` man pages.

ncm-cron

This component configures the cron daemon. It controls the entries in the `/etc/cron.d` directory. Entries managed by this component will have the `ncm-cron.cron` suffix. For more information read the `ncm-cron(8)` and the `cron(8)` man pages.

ncm-ntpd

This component configures the Network Time Protocol daemon to be started at run time. It reads from `/system/network/timeservers` and modify the `/etc/ntp.conf` file. For more info read `ncm-ntp(8)` and `ntpd(8)` man pages.

C2. OPTIONAL CONFIGURATION COMPONENTS

The optional configuration components are listed in Table 3. They should be included in the template

```
pro_system_mycluster
```

together with those generic configuration parameters related to your cluster. (see Section 5.7. for more information). It is up to the system manager to decide which components to use. By default, none of those configuration components is included.

Component	Description
ncm-access_control	configures access control by modifying several system files
ncm-altlogrotate	alternate logrotate configuration component
ncm-authconfig	component to manage system authentication services
ncm-autofs	a component to configure the autofs service
ncm-ccm	a component which writes the ccm configuration file
ncm-cdp	a component which configures the cdb-listend configuration file
ncm-chkconfig	component for runlevel management via chkconfig
ncm-dirperm	permissions and file/directory creation
ncm-etcservices	/etc/services configuration component
ncm-filecopy	for installing generic configuration files
ncm-fmonagent	configures the Lemon fmon-agent software
ncm-iptables	component for IPTABLES firewall setup
ncm-ldconf	component to manage /etc/ld.so.conf file
ncm-lmsensors	configures /etc/rc.d/rc.local (loads I2C chip drivers)
ncm-logrotate	component to control the logrotate configuration
ncm-mailaliases	configures the correct root entry in /etc/aliases
ncm-netdriver	configuration for network card driver module
ncm-network	for networking configuration on systems based on /etc/sysconfig
ncm-nfs	component to control entries in the /etc/exports and /etc/fstab files for NFS
ncm-portmap	configures the portmapper to start at boot time
ncm-profile	creates profile scripts for setting environment variables and paths
ncm-rproxy	configures Apache 2.0 to run as a reverse caching proxy
ncm-serialclient	configures /etc/inittab and /etc/securetty to allow root access and a getty process over the serial line
ncm-smartd	component for SMARTd (Self Monitoring Analysis Reporting daemon) configuration
ncm-ssh	configures ssh on a node
ncm-state	configures /etc/state.desired (desired production state)
ncm-sysctl	configures /etc/sysctl.conf (configure kernel parameters at runtime)
ncm-yaim	component for configuring LCG-2 via YAIM

Table 3: Optional Configuration Components

D INSTALLATION USING APT AND YUM

Quattor does not impose using any package distribution tool. Even if SPMA is part of the Quattor core distribution, it is possible to use alternative mechanisms for software distribution. Wide-spread examples available for Scientific Linux are APT and YUM.

D1. SPMA VERSUS APT AND YUM

Both SPMA and APT/YUM have advantages and disadvantages.

Advantages using SPMA:

- SPMA is declarative: The list of packages which should be on a cluster/node is defined in CDB. SPMA behaves stateless and will upgrade/downgrade/install/remove packages as required on the cluster nodes, in order to match the CDB target configuration.
- Every node may have a completely different setup in terms of installed packages and their versions. There is no need to go for the latest packages found in a repository (nor to create a new repository everytime a new combination of package versions is to be tried out).
- SPMA and SWRep provide an explicit separation of package *depot* (SWRep) and *configuration* (in CDB). You can add multiple versions (old, production, new, beta) of packages in the repository, this won't change the client setup until you modify the corresponding CDB templates.
- SPMA supports rollbacks. Just change the packages/versions declared in CDB and SPMA will take care to downgrade/upgrade/install/remove whatever is required.
- SPMA supports installing multiple simultaneous versions of the same package.

Advantages using APT or YUM:

- APT and YUM are standard tools in particular when using Scientific Linux distributions.
- APT repositories are available for many software sets (including LCG, and of course Quattor)
- APT and YUM do dependency resolution - dependent packages are automatically resolved and included in the list of packages to be installed (note that however, this requires a careful maintenance of repository contents as otherwise, a different set of packages fulfilling the dependency requirements could be pulled instead of the desired list).

The choice between SPMA and APT/YUM will depend on the complexity of your environment and/or the control level you need. As an example, at CERN the SPMA is used for managing the Computer Centre batch nodes and servers, whileas APT is used for desktops.

APT (or YUM) and SPMA should not be used in parallel on the same node. However, it is possible (and very confortable) to perform the initial bootstrap of the Quattor server with APT, and then use SPMA to manage your nodes!

D2. HOW TO SET UP APT/YUM FOR QUATTOR

APT and YUM repositories containing the Quattor packages are found at the following URL[22]:

APT: `http://quattorsw.web.cern.ch/quattorsw/software/quattor/apt`

YUM: `http://quattorsw.web.cern.ch/quattorsw/software/quattor/yum`

[22] As of release 1.1.X, the APT and YUM repositories are identical

APT configuration

Create a file called `/etc/apt/sources.list.d/quattor.list` with the following contents:

```
rpm http://quattorsw.web.cern.ch/quattorsw/software/quattor apt/1.1.X/i386 quattor_sl3
```

Then run (as root) `apt-get update`. You are ready to install packages using `apt-get install <packagename>`.

Important note: If the APT's preference file `/etc/apt/preferences` defines a default/alternative repository holding different versions of some requested packages, you might experience dependency errors. Please check the above file, and rename it, if needed.

YUM configuration

Add to your `/etc/yum.conf` configuration file the following:

```
[quattor]
name=quattor
baseurl=http://quattorsw.web.cern.ch/quattorsw/software/quattor/yum/1.1.X/i386/RPMS.quattor_sl3
```

You are ready to install packages with the command `yum install <packagename>`.

D3. APT/YUM META-PACKAGES FOR QUATTOR

For each Quattor subsystem, a meta-package RPM (package which contains no files but dependencies on other packages) has been created. They are the following:

- `quattor-client`: Install client packages (CCM, NCM + basic components, CDB CLI, SWRep CLI)
- `quattor-cdb`: Install Configuration Database (CDB) server
- `quattor-cdbsql`: Install the CDBSQL backend server
- `quattor-aii`: Install the Automated Installation Infrastructure (AII).
- `quattor-swrep`: Install the SPMA Software Repository (SWRep) server
- `quattor-spma`: Install the SPMA Software Package Manager Agent.[23]

The meta-packages are installed using `apt-get install quattor-<x>` (APT) or `yum install quattor-<x>` (YUM).

D4. ADDITIONAL STEPS

Configuring subsystems

Apart of installing the RPM's, you still need to configure the subsystems as described in this guide! We recommend you to install and configure the subsystems following the order defined in chapters 2 to 8.

In particular, you should generate a profile excluding the `pro_software...` templates, as described in 5.9.. (But you should not comment out the SPMA component, as many other components have a dependency on it - the NCM spma component will **not** run SPMA if there is no SWRep software repository defined, but just exit with an info message).

Changes to default AII KickStart template

You also need to change the AII KickStart template to use APT/YUM instead of SPMA for bootstrapping. The default AII KickStart template contains documentation on how to do this.

[23] note that as stated before, SPMA should not be used simultaneously with APT or YUM, but these can be used for bootstrapping SPMA of course.

E QUATTOR FOR LCG-2 INSTALLATIONS

With Quattor it is possible to install the LHC Computing Grid (LCG) software release called LCG-2. There are two main possibilities.

E1. INSTALLATION USING NCM-YAIM

The simplest way, but less powerful and flexible to install LCG-2 is using the YAIM component (ncm-yaim) which sits on top of YAIM, the LCG-2 configuration script. ncm-yaim generates a YAIM configuration file out of information stored in CDB, and invokes yaim. ncm-yaim can be used with APT/Yum or SPMA.

See the ncm-yaim(8) man page for more details on this component.

E2. INSTALLATION USING THE QUATTOR WORKING GROUP COMPONENTS AND TEMPLATES

Another possibility is to use the templates and components produced by the LCG PEB Quattor Working Group (QWG). This probably requires more work but in return allows a more fine grained configuration than using YAIM.

- In addition, the QWG templates are an excellent example of how to build a complete set of templates.

Please consult [14] for more information on the QWG.

F UPGRADING AN EXISTING QUATTOR ENVIRONMENT

This section describes the required steps for upgrading an already existing Quattor environment (e.g. running Quattor 1.0).

F1. SERVER

In theory, it should be possible to upgrade the Quattor server software (AII, SWRep, CDB) directly on the server(s) running an older Quattor release. However, in practice, this has not been sufficiently tested, and we discourage this unless you know exactly what you are doing. We recommend to set up new server node instances and to install the server software from scratch.

CDB migration

For setting up the new CDB server, you first need to export all existing templates from your old server. You can do the export using `cdbop --command "get *"` (see the cdbop man page for details).

However, the import is unfortunately a manual process. You have to take into account that many default templates have changed from 1.0 to 1.1. You should not overwrite templates belonging to the 'standard' set with their old equivalents. The site specific templates (host, software packages, cluster templates) also need to be adapted as some definitions have been updated.

SWRep migration

For migrating SWRep, we recommend to do the following for each platform: directly copy over the RPM's to the new server and to run the `bootstrap` command of `swrep-client`.

Alternatively, and if you have many SWRep 'areas' defined for each platform which you would like to preserve, you can also copy the *complete* contents of /var/www/html/swrep to the new server. This will make a one-to-one copy of your old SWRep server. This procedure is guaranteed to work for Quattor 1.0 and 1.1, but may be obsoleted in the future and replaced by a more encapsulated mechanism.

F2. CLIENTS

There is no reinstall required on the client nodes, and it is possible to migrate from the old to the new server, by just reconfiguring the server location in the CCM, via the `ncm-ccm` component, and to update the CDB location in the AII options. The relevant entries are:

```
/software/components/aii/osinstall/options/cdb
/software/components/ccm/profile
```

Everything else is derived out of the configuration information in the configuration profile.

You need to ensure that prior to migration, your clients have **both** `ncm-spma` 1.2.11 or higher and `ncm-ccm` 1.0.0 or higher installed (and activated!).

Switching the server location (via `ncm-ccm` which updates /etc/ccm.conf) is always risky, and a bad configuration here may leave the clients pointing into Nirvana! Safety checks on ncm-ccm for avoiding this situation are planned for but not available yet (see Savannah #9755).

F3. USING DNS ALIASES

In order to minimize the risk of misconfigurations, we recommend to define DNS alias(es) for the quattor server(s), and to point the clients towards these instead of pointing them directly to physical nodes. The DNS aliases can then be updated in case of server changes, updates or upgrades - and quickly reverted in case of problems.

A single DNS alias is sufficient if you run all the quattor services on one server (e.g. `quattorsrv.mysite.org`). Otherwise, you should define a DNS alias for each service (`swrep.mysite.org`, `cdbserv.mysite.org`).

For the test phase prior to a full migration, it is possible to define a new alias (`quattorsrvtest.mysite.org`) and migrate some test nodes to it prior to change the general DNS alias(es).

G FIREWALL FOR QUATTOR SERVER

On a quattor services containing all services, then the following ports must be open:

- http:tcp
- https:tcp
- ssh:tcp
- dhcp:tcp
- tftp:udp
- 7777:udp

These allow the clients to access their machine profiles via the web server and to boot via the AII subsystem. These can be opened either manually or using the firewall configuration tool.

For TFTP to work, a further range of temporary udp ports must be opened to allow clients to retrieve files. For example, to open ports 30000-35000 for TFTP, run the following commands as root:

```
# iptables -I input 1 --destination-port 30000:35000 -p udp -j ACCEPT
# echo 30000 35000 > /proc/sys/net/ipv4/ip_local_port_range
# service iptables save
```

This will set the kernel to use only the given port range for temporary tcp and udp ports (make sure you configure a range large enough for your load) and allow those udp ports to be contacted. The same can be done on older RedHat-like linux distributions, such as RH7.3, replacing the `iptables` command with `ipchains`.

G1. CONFIGURING IPTABLES VIA NCM

Note that a NCM component for configuring `iptables` exists and is part of the Quattor distribution. You can find more information in the `ncm-iptables` man page.

H MANAGING THE CDB WITH CDBOP

With the `cdbop` client interface you have full control, interactively, over your CDB contents. Here we describe some typical management sessions, involving adding, deleting and editing configuration templates.

A `cdbop` session is similar to an ftp session, that is, you need first to log into the server, then you may download (check out) some templates of interest, possibly modify them, lastly you upload the modified files (commit) and log out. Since all modifications to template files are done on local copies, it is a good practice to run the CDB client inside a buffer directory where files can be freely downloaded and removed.

H1. POPULATING THE CDB

Right after the CDB installation, your database is empty, thus, the first thing to do is to populate it with a set of templates. When you run `cdbop` a session is automatically opened:

```
$ cdbop
quattor CDB CLI: Version 1.8.14

Enter user-name: admin
Enter password:

Connecting to http://mycdbserv...
Welcome to CDB Command Line Interface
Opening session...
[WARN] can not list the templates from CDB

Type 'help' for more info

[cdb] ~ > help
Type 'help command' for more detailed help on a command.
  Commands:
    add      - add new templates
    close    - close user session if there are no uncommitted user ...
    commit   - perform compilation and validation of HLD and ...
    delete   - delete given templates
    eval     - how to run external commands
    exit     - quit cdb command line interface
    get      - download private (local) copy of source templates
    help     - prints this screen, or help on 'command'
    lcd      - change local working directory
    list     - list the templates
    open     - open user session on CDB
    rollback - discard all the changes introduced since beginning ...
    update   - update existing templates
    ...
```

The warning message reported after the session opening is normal because the CDB is empty. Now, suppose your templates are installed in the directory `/opt/my-pan-templates`, then, you can move there with the `lcd` command and upload some new templates with the `add` command:

```
[cdb] ~ > lcd /opt/my-pan-templates
[cdb] /opt/my-pan-templates > !ls
pro_declaration_functions_filesystem
pro_declaration_functions_general
pro_hardware_asus_terminator_p4_533a.tpl
```

```
profile_mynode.tpl
...
[cdb] /opt/my-pan-templates > add *.tpl
[INFO] adding template: pro_declaration_functions_filesystem
[INFO] adding template: pro_declaration_functions_general
[INFO] adding template: pro_hardware_asus_terminator_p4_533a.tpl
[INFO] adding template: profile_mynode.tpl
...
[cdb] /opt/my-pan-templates > commit
[cdb] /opt/my-pan-templates > list
pro_declaration_functions_filesystem
pro_declaration_functions_general
pro_hardware_asus_terminator_p4_533a.tpl
profile_mynode.tpl
...
```

Notice that it is possible to run external commands by preceding them with a '!' sign. In the example above, `!ls` is called to list the content of the directory `/opt/my-pan-templates`, whereas `list` gives the content of the CDB. Also, shell-like patterns can be used with all commands involving templates.

At this point the CDB contains some templates, and you can start to adapt them to your needs.

H2. MODIFYING A TEMPLATE

Choose a buffer directory where to download the template(s) you want to edit. Use the `get` command to check out a template (a local copy is created). Then edit the file, `update` it and `commit` the changes. `cdbop` has no facility for file editing, hence you have to call your favorite editor.

```
[cdb] /opt/my-pan-templates > lcd ~/tmp
[cdb] ~/tmp > get profile_mynode
[INFO] getting template: profile_mynode
[cdb] ~/tmp > !emacs profile_mynode.tpl &
[cdb] ~/tmp > update profile_mynode.tpl
[INFO] updating template: profile_mynode
[cdb] ~/tmp > commit
```

Note that `emacs` is launched in the background. After the commit, any template local copy is still in the buffer directory: if you try to check out again one of them, an error is reported; the solution is to remove the local copy, as in the following example:

```
[cdb] ~/tmp > !ls
profile_mynode
[cdb] ~/tmp > get profile_mynode
[ERROR] file 'profile_mynode' already exists
[cdb] ~/tmp > !rm profile_mynode.tpl
[cdb] ~/tmp > get profile_mynode
[INFO] getting template: profile_mynode
```

Once you have committed all the changes to the CDB, you can safely quit with the command `exit`. Warning: any uncommitted change will be lost without any advice!

H3. OTHER USEFUL COMMANDS

Other commonly used commands in `cdbop` are the following:

- `delete`: remove one or more templates from current session. It does not remove files from current buffer directory (use `!rm` instead);
- `rollback`: discard all the changes introduced since the beginning of the transaction, that is, undo all updates added to the current session since the last commit;
- `validate`: validates changes introduced during the current session without committing them.

For the complete list of all supported commands see the `cdbop(1)` man page[24].

[24] Currently, advanced commands for access control management are still under development on the CDB server side

I SECURE TRANSPORT OF CONFIGURATION INFORMATION

In case CDB is used for holding sensitive information, the contents of the XML configuration profiles should not be disclosed. This goal can be achieved by adopting HTTPS as the unique protocol for the distribution of such files. As a reminder, the system is based on the following mechanisms:

1. XML profiles are by default transferred from CDB to CCM using HTTP: in case the XML profile holds sensitive data (such as login names and ciphered passwords), this transfer mode is insecure.
2. No authentication procedure between clients and the server is in place, which exposes the system to attacks based on spoofing the name either of the server or of the client. Possible attack scenarios are those wherein a spoofed server provides the clients with false/corrupted profiles, or wherein a spoofed client obtains profiles related to other nodes.
3. The CCM caches the profiles as DBM files with world-readable permissions: this is potentially dangerous as it could disclose sensitive information to unauthorized users of the client node if sensitive information is stored in CDB.

Quattor supports HTTPS for secured profile transfer. In case sensitive information is to be stored in CDB, the following changes are recommended:

1. switch to HTTPS for XML profile distribution: this requires to modify some CCM configuration parameters (Section I3.), and to add some Apache directives as well (Section I5.);
2. use non-world/group-readable permissions on the CCM side (Section I3.).

I1. ADVANTAGES OF USING HTTPS WITH CCM

Using HTTPS with CCM provides several advantages:

- secure information transfer, since data are encrypted: this prevents eavesdroppers from obtaining information in transit over the network;
- client to server authentication and vice versa: on one hand, this allows to enforce access policies to sensitive data according to the client "name", on the other hand, clients are guaranteed to talk to the original server.

The secure transfer is achieved through public/private cryptography, and the authentication is based on X.509 certificates. A *certification authority* (CA) is needed for certificate validation: since we need validation only within one organization, it might be sufficient to set up a CA locally, e.g. on the CDB server. However, for better security, you could resort to a hierarchy wherein, at each level, a CA provides guarantees for the underlying level, and in its turn receives guarantees from the upper level. Since administering a CA hierarchy is a non trivial task, in this document, we show only how to set up a single CA which uses a self-signed certificate.

I2. HOW TO GENERATE YOUR OWN KEYS AND CERTIFICATES

Here we describe a way to generate keys and certificates for both the client and the server side of the system (see [15]). Even though the following procedure is based upon standard Unix/OpenSSL commands, it requires manual intervention on all the clients involved, which is clearly infeasible on a large computer farm: we are currently working on a mechanism to automate the creation and distribution of keys and certificates for several client nodes. For the sake of simplicity, all necessary files are generated on one node (the CDB server, for instance) and then transferred to the right places.

Setting up a local Certification Authority

A local CA may be set up on the CDB server node. Log on to it, choose a suitable directory, move there and run the following commands (you need not to be root; `cdb$` is the prompt for a non-privileged user on the CDB server).

Generate an encrypted private key for the CA (remember the passphrase you will use):

```
cdb$ cd <your-CA-directory>
cdb$ openssl genrsa -des3 -out ca.key 1024
```

Check that the key file is not world-readable.

Generate a self-signed certificate (unless you have a higher CA that can sign your certificate):

```
cdb$ openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```

The `ca.crt` is the CA's certificate to be used for verifying the signature of a peer that wants to be authenticated.

In order to simplify the signing of a "certificate signing request" (CSR), you may use the `sign.sh` script distributed with the `mod_ssl` [16] for Apache 1.x, in the sub-directory `pkg.contrib`. Just copy it in `<your-CA-directory>` and make it executable, if necessary.

Creating client/server files

First, create a CCM client's private key, from within the same `<your-CA-directory>` on the CDB node:

```
cdb$ openssl genrsa -out ccm.key 1024
```

This command should return the file `ccm.key`. Note that this key is *clear text*, hence it is paramount not to diclose it to normal users: only root should have permission to read it.

Then, create a CSR:

```
cdb$ openssl req -new -key ccm.key -out ccm.csr
```

You could avoid answering questions by writing a small configuration file: see the OpenSSL 'req' man page for further details.

Lastly, the CSR file `ccm.csr` is signed by the local CA[25]:

```
cdb$ ./sign.sh ccm.csr
```

You should get a file `ccm.crt` which is the certificate the client must provide when requested for authentication[26].

Now, you may want to use a secure mechanism such as `scp` to transfer the client's private key and signed certificate, and the CA certificate to the CCM node (currently you need root access to the client node):

```
cdb$ scp ccm.key ccm.crt ca.crt root@cdb:<your-ccm-config-dir>
```

where `<your-ccm-config-dir>` is usually `/etc/`.

As for the server's key and certificate, repeat the above procedure to generate the files `cdb.key`, `cdb.crs` and `cdb.crt`. This time you need no network transfer since the CA and the CDB server reside on the same machine, but the created files must be copied in their right place by the superuser (note the change in the prompt `cdb#`), because they will be used by the Apache server:

[25] In the real world it would be sent to a trusted CA, which would return it signed to us.

[26] This file holds the client's public key indeed.

```
cdb$ openssl genrsa -des3 -out cdb.key 1024
...
cdb$ openssl req -new -key cdb.key -out cdb.csr
...
cdb$ ./sign.sh cdb.csr
...
cdb$ su -
...
cdb# cp cdb.key <your-cdb-ssl-key-dir>
cdb# cp cdb.crt <your-cdb-ssl-crt-dir>
cdb# cp ca.crt <your-cdb-ssl-ca-dir>
```

Remember the passphrase you entered when generating the encrypted key! Note that, since the Apache server is adopted on the CDB node, different directories are usually dedicated to SSL-related files: for instance, `/etc/httpd/conf/ssl.key`, `/etc/httpd/conf/ssl.crt` and `/etc/httpd/conf/ca`, respectively. If these directories do not exist, make sure your Apache's SSL module is installed (see Section I5.), and create them.

Now that all needed objects are ready, we can step into the configuration phase of the CCM client and of the CDB server.

I3. HOW TO ENABLE CCM TO USE HTTPS AND CERTIFICATES

To enable HTTPS in CCM you need three files, generated according to the procedure seen in I2.:

1. a client private key: `ccm.key`;
2. a client certificate holding and validating its public key: `ccm.crt`;
3. a trusted CA certificate: `ca.crt` [27].

The first two files are used for client authentication toward the server, the third allows to verify the server identity.

Some parameters need to be set or modified in the CCM configuration file `/etc/ccm.conf` (you must log in to the client node as root):

- `key_file`: absolute file name of the client's private key file;
- `cert_file`: absolute file name of the client's certificate file;
- `ca_file`: absolute file name of the file holding either a self-signed certificate from a trusted CA or a "certificate chain".

Since more than one CA certificate might be used for server verification, the last parameter (`ca_file`) can be substituted with the `ca_dir` parameter which specify the directory containing the CA certificates. But in this case, for each certificate, you need to create a soft link whose name is the hash of the linked certificate file name (see Section I3.2.).

Sample configuration with a single trusted CA certificate file

The CCM configuration file `/etc/ccm.conf` holds the following snippet (remember to copy the needed files in the right place!):

```
key_file  /etc/ccm.key
cert_file /etc/ccm.crt
ca_file   /etc/ca.crt
```

[27] There is need to change the CCM documentation as it mentions a "server certificate" instead of a "CA certificate" which might be misleading.

Make sure that the `/etc/ccm.key` file is not world-readable. As for CCM cache files, the same is obtained with this last configuration parameter:

```
world_readable 0
```

Sample configuration with multiple trusted CA certificates

The CCM configuration file `/etc/ccm.conf` holds the following snippet:

```
key_file  /etc/ccm.key
cert_file /etc/ccm.crt
ca_dir    /etc/ca/
```

Check that the `/etc/ccm.key` file is not world-readable. Now suppose you trust two different CA's, hence the `/etc/ca/` directory must hold two certificate files (one of them should match the CDB server certificate!):

```
ccm# cd /etc/ca; ls
ca1.crt
ca2.crt
```

The above files may contain either a single self-signed certificate or a certificate chain. Then, to create the hash links, use the `Makefile.crt` provided with the Apache's `mod_ssl` distribution in the directory `/etc/httpd/conf/ssl.crt/` (see also Section I5.); just place it in the directory `/etc/ca` and issue:

```
ccm# make -f Makefile.crt
...
ccm# ls -l
total 12
-rw-r--r--  1 root   root       1322 Mar  2 16:55 ca1.crt
-rw-r--r--  1 root   root       1322 Jan 26 15:30 ca2.crt
lrwxrwxrwx  1 root   root          7 Mar  2 16:55 d2227077.0 -> ca1.crt
lrwxrwxrwx  1 root   root          6 Mar  2 16:55 9sdf4335.1 -> ca2.crt
-rw-r--r--  1 root   root       1522 Mar  2 15:17 Makefile.crt
```

The above operation must be repeated each time you change your certificates. You may notice that all the above files are owned by the *root* user: this is due to the fact the all CCM applications are owned by the superuser and run with its privileges. Future releases of CCM will have a dedicated non-privileged user for all its duties.

Avoid world-readable CCM cache files as seen above.

I4. HOW TO ENABLE CDB-SYNC TO USE HTTPS AND CERTIFICATES

Since the *Notification Client Handler* in the CDB synchronization mechanism needs to access protected data, the `cdsync-nch` application has been extended to support HTTPS as well.

To configure `cdsync-nch` just edit its configuration file `/etc/cdsync.conf` the same way as seen for CCM in Section I3..

I5. WHAT NEEDS TO BE CHANGED ON THE CDB SERVER SIDE

Since on the CDB side Apache 2.0 is adopted as a file server, two dedicated modules `mod_ssl` and `mod_rewrite` can be used. Make sure that they are installed on your system: they should be listed in `/usr/lib/httpd/modules` as `mod_ssl.so` and `mod_rewrite.so` files. On SLC3, the former, used for all SSL-based transactions, is usually shipped in a separate package `mod_ssl-<ver>.<arch>.rpm`, whereas the latter, used for filtering the requests, comes from the standard Apache distribution; other distributions may ship both modules in the same package.

The basic idea is to set up a filtering mechanism that protects the access to the XML profiles in this way:

- XML profiles are accessed only via HTTPS;
- a standard client node *foo* can access *only* its `profile_foo.xml` file;
- some privileged client nodes (e.g. those belonging to an "administration cluster") can access *all* the XML profiles.

Directives in the main Apache's configuration file

The main Apache's configuration file `/etc/httpd/conf/httpd.conf` must contain the following directives:

```
LoadModule      rewrite_module modules/mod_rewrite.so
Include         conf.d/*.conf
HostnameLookups On
```

The first of the above two lines allows to load the `mod_rewrite` modules, and the second line instructs Apache to parse all module-related configuration files listed in the directory `/etc/httpd/conf.d/` as `<module-name>.conf` files.

Directives in the `mod_ssl`'s configuration file

The changes to the file `/etc/httpd/conf.d/ssl.conf` are somewhat more complex (for a better understanding see [17, 18]). First of all, check that the following directive exists:

```
LoadModule ssl_module modules/mod_ssl.so
```

Then, inside the `<VirtualHost...>` context add/modify the following directives to match your system set-up (other default settings should work as they are):

```
DocumentRoot          "<path-to-https-www-root>"
...
SSLCertificateFile    <your-cdb-ssl-crt-dir>/cdb.crt
SSLCertificateKeyFile <your-cdb-ssl-key-dir>/cdb.key
SSLCACertificateFile  <your-cdb-ssl-ca-dir>/ca.crt
...
# Directives to restrict access to XML profiles
<Directory "<path-to-https-www-root>/profiles"
    Options         +Indexes
    SSLOptions      +StdEnvVars
    SSLVerifyClient require
</Directory>
RewriteEngine on
RewriteMap    ACLmap txt:<path-to-your-map-dir>/ACLmap.txt

RewriteCond   ${ACLmap:%{REMOTE_HOST}|NO} NO
RewriteRule   ^/.*$ /profiles/profile_%{REMOTE_HOST}.xml

RewriteCond   ${ACLmap:%{REMOTE_HOST}|NO} NO
RewriteRule   ^/profiles/profile_(.*)<dot-esc-dname>\.xml$ \
                  /profile_$1.xml [L]
```

The <path-to-https-www-root> area is accessible only via HTTPS: it has a profiles/ directory entry, which is the XML repository we want to protect from unauthorized access. It is mandatory that <path-to-https-www-root> be a separate directory, with respect to the ordinary HTML file repository, defined by the *DocumentRoot* directive inside the main Apache's configuration file! Thus, if you have a simple HTTP-based CDB installation serving XML files, for example, from /var/www/html/profiles/, you must move the profiles/ directory to a different place, like /var/www/https/ (usually, it is a matter of changing a symbolic link).

The three directives starting with *SSL*... define, respectively, the server's certificate, the server's private key and the CA's certificate: these are the files created in Section I2.. Since more than one certificate may be present, for both the server and the CA, hash symlinks should be created as seen in I3.2.:

```
cdb# cd <your-cdb-ssl-crt-dir>
cdb# make -f Makefile.crt
...
cdb# cd <your-cdb-ssl-ca-dir>
cdb# make -f Makefile.crt
```

Inside the <Directory...> context, it is stated to require authentication for each client connecting.

Lastly, we have a set of mod_rewrite directives which define a selective filter based on an "access control list" (ACL) stored in the text file indicated by the RewriteMap line. This map must hold the long hostnames (that is, fully qualified domain names) of the privileged nodes which have access to all the XML profiles on the server[28]. An example of ACL could be the following:

```
lxadm00.<your-domain>    YES
lxadm01.<your-domain>    YES
...
lxadm09.<your-domain>    YES
```

The ACL must also include the hostnames of clients connecting to the CDB server through other programs, such as cdbop and cdb-simple-cli.

Indeed, the last three lines given for the ssl.conf file act as the following procedure:

- check the first rule: whatever URI is requested look up the ACLmap:
  - if the remote hostname is tagged 'NO' or it is not listed then rewrite the URI as 'profile_*remote-hostname*.xml', else do nothing;
- check the second rule: if the URI is in the form 'profile_*foo*.cern.ch.xml' rewrite it as 'profile_*foo*.xml' (unless we are a privileged client listed in the ACLmap).

In other words, if a non-privileged hostname *bar* requests a file which is not its profile definition, the URI is rewritten in 'profile_*bar*.xml', so that no other file can be obtained by a possibly cheating client; conversely, privileged clients are allowed to get whatever file they request. The second rule is necessary because the first one uses the variable 'REMOTE_HOST', which is always expanded to the fully qualified domain name (e.g. 'lxplus001.cern.ch'), whereas our file names contain only the short hostname (i.e. 'lxplus001'); note that you must replace <dot-esc-dname> with your domain name, having care of escaping all the dots as in \.mydomain\.net.

Your CDB server is now configured! Remember to restart Apache to have it work correctly: you will be prompted for the passphrase used to protect the cdb.key. Once your server is properly secured (double-check file and directory permissions!), you may remove the encryption with the command

```
cdb# openssl rsa -in cdb.key -out cdb.cleartext.key
```

and do not forget to update the Apache SSL configuration. No passphrase will be asked any more.

[28] Optionally, non-privileged hostnames can be listed as well and tagged with 'NO', but this would slow down the look-up.

J PROXY-CACHING DEPLOYMENT WITH APACHE

As seen in Section 4.9., it is possible to install one or more proxy-caching servers in order to speed up package downloads between the client nodes and the software repository (SWRep). This section deals with the basic steps to set up and configure a single cache based on the Apache HTTP Server Version 2.0; installations based on Apache 1.3 are also possible but no automated deployment support is available in Quattor.

The packages/components needed for a fully automated deployment are:

- Apache 2.0:
  `httpd-<ver>.<arch>.rpm`;
- a garbage collector program, such as `htcacheclean` which is distributed with the source code of Apache 2.1. A special package for Quattor is available:
  `proxy-htcacheclean.<ver>.i386.rpm`;
- an NCM component to properly configure Apache:
  `ncm-rproxy-<ver>.noarch.rpm`;
- the optional NCM component to run the garbage collector via cron:
  `ncm-cron-<ver>.noarch.rpm`.

Additionally, the proxy support in SPMA must be enabled: see the spma(1) man page for details.

J1. REVERSE PROXY-CACHE CONFIGURATION

Suppose you want to install a proxy-cache on a machine called `proxy.your.domain`, for intercepting requests to a backend server called `backend.your.domain`. Apache 2.0 can be configured as a reverse proxy-cache with the `ncm-rproxy` component. Basically, this modifies the main configuration file (`httpd.conf`) to enable support for proxy and cache functionalities, and adds a configuration file (`rproxy.conf`) dedicated to specific proxy-caching directives. For a complete overview about Apache's proxy-caching options see [19, 20].

Two configuration template files come with ncm-rproxy:

```
pro_software_component_rproxy.tpl
pro_declaration_component_rproxy.tpl
```

Your proxy-node's profile needs only to include the former since this, in its turn, includes the latter. Some mandatory settings must appear somewhere, after the 'include' statement for the above template. First, restrict the access to your proxy from a specific domain only; for instance, the following line allows only requests coming from `.your.domain` (note the leading dot) to access the proxy:

```
"/software/components/rproxy/proxy/domain" = ".your.domain";
```

Then, define the mappings between downloadable objects and backend repositories; for instance, these two lines map all requests rooted at `/swrep` onto the server reachable at `http://backend.your.domain/swrep`:

```
"/software/components/rproxy/proxy/mappings/0/path" = "/swrep";
"/software/components/rproxy/proxy/mappings/0/url" =
    "http://backend.your.domain/swrep";
```

Remember that all client applications wishing to use the proxy must be configured to send requests to `http://proxy.your.domain`. Once your CDB is updated with the above template(s), activate the new configuration on your proxy node by issuing

```
# ncm-ncd --configure rproxy
```

`ncm-rproxy` has also an 'unconfigure' option to restore the Apache configuration in effect before the last 'configure' action. See the ncm-rproxy(1) man page for details.

Next, if you care about disk room consumption on your proxy node, it is recommended to install a garbage collection program. This is not strictly necessary, but you have to be sure that the proxied content is bounded and can fit in your disk! If you choose to use `proxy-htcacheclean`, you may either run the program as a daemon, through the `-d` option, or via a cron job: in the latter case, hints on how to configure cron with `ncm-cron` are given in the README file distributed with the package. The README file provides also detailed instructions on how to build `htcacheclean` from the Apache's source code.