Introduction to Pan Core Templates

Rafael A. García Leiva
angel.leiva@uam.es
Department of Theoretical Physics
Universidad Autónoma de Madrid

Geneva – January 2005

UAM

quattor

Contents

- Introduction
- Standard Templates
- Site Specific Templates
- Configuration Components Templates
- Clint Node Templates
- Pan User Conventions Guide

The Set of Pan Templates

Goal:

- To provide a set of templates for quattor developers and testers.
- To provide an example of pan templates for the installation and configuration of a prototype small site.
- To provide an starting point to develop a new customized set of templates adapted to our needs.

What it is not:

- A set of templates for the installation and configuration of a LCG (DataGrid, EGEE, ...) site.
- A set of templates that can be used for all kind of installations: clusters, desktop machines, general fabric management, grid computing ...

Organization of the Templates

Example Templates:

`/usr/share/doc/pan-templates-1.1.4`

directory: `standard`

- Common templates to most of the environments
- There is no need to change them

directory: `site_specific`

- Templates with specific information to local environment
- System managers should change them

directory: `components`

- Configuration components templates
- Use only those templates you need

Tutorial Templates:

They should be already loaded on your own CDB

UA

quattor

Standard Templates

Common Data Types (1/2)

Templates:

- `pro_declaration_types`
- `pro_declaration_type_validation_functions`

Datatype	Description
`date`	date and time (deprecated, do not use)
`ansdate`	date and time format consistent with LDAP
`isodate`	date and time format consistent with W3C
`hwaddr`	NIC MAC address
`ipv4`	numerical IPv4 address
`ipv6`	numerical IPv6 address
`ip`	represents either an Ipv4 or IPv6
`fqdn`	fully-qualified domain name

Common Data Types (2/2)

Templates:

- `pro_declaration_types`
- `pro_declaration_type_validation_functions`

Datatype	Description
`hostname`	hostname (either a fqdn or an IP)
`shorthostname`	short hostname
`hostport`	host and port in the form host:port
`URI`	Uniform Resource Identifiers
`absoluteURI`	absolute URI
`hostURI`	host URI
`email`	email address

Common Units

Template:

- `pro_declaration_units`

Unit	Description
`mb` or `MB`	1
`gb` or `GB`	1024 * mb
`tr` or `TB`	1024 * gb
`MHz`	1
`GHz`	1000 * MHz

Common Structures

Templates:

- `pro_declaration_structures`
- `pro_declaration_structure_validation_functions`

Provides support for the profile's tree organization:

- `annotation`
- `ram`
- `harddisk`
- `cpu`
- `nic`
- `cards`
- `hardware`
- `interface`
- `network`
- `software`
- `system`
- `component`
- `profile`

Declaration of Profile's Base

Template: `pro_declaration_profile_base`

```
type "/" = structure_profile;
```

Template: `pro_declaration_structures`

```
define type structure_profile = {
  "hardware" : structure_hardware
  "system"   : structure_system
  "software" : SOFTWARE
};
```

Declaration of Profile's Base

Template: `pro_declaration_structures`

```
define type structure_hardware = {
  include structure_annotation
  "cpu"           : structure_cpu[]
  "harddisks"     : structure_harddisk{}
  "ram"           : structure_ram[]
  "cards"         : structure_cards
};
```

Template: `pro_declaration_structures`

```
define type structure_system = {
  "network"     : structure_network
  "cluster"     : structure_cluster
  "kernel"      : structure_kernel
  "filesystems" : structure_filesystems
  ...
```

Declaration of Profile's Base

Another view of profile's tree (`ncm-query --dump`):

```
+-/
  +-hardware
  | +- cpu
  | |  +- 0
  | |  | $speed : string
  | |  +- ...
  | +- harddisks
  | +- ram
  | +- cards
  +-system
  | +- network
  | +- cluster
  | +- kernel
  | +- filesystems
  | ...
```

Common Utility Functions (1/2)

Template:

- `pro_declaration_functions_general`
- `pro_declaration_functions_network`
- `pro_declaration_functions_filesystem`

Function	Description
`default`	Return either current value for a path if it exists and is defined or the given default value
`push (nlist)`	Push a value onto the end of a list (nlist)

Common Utility Functions (2/2)

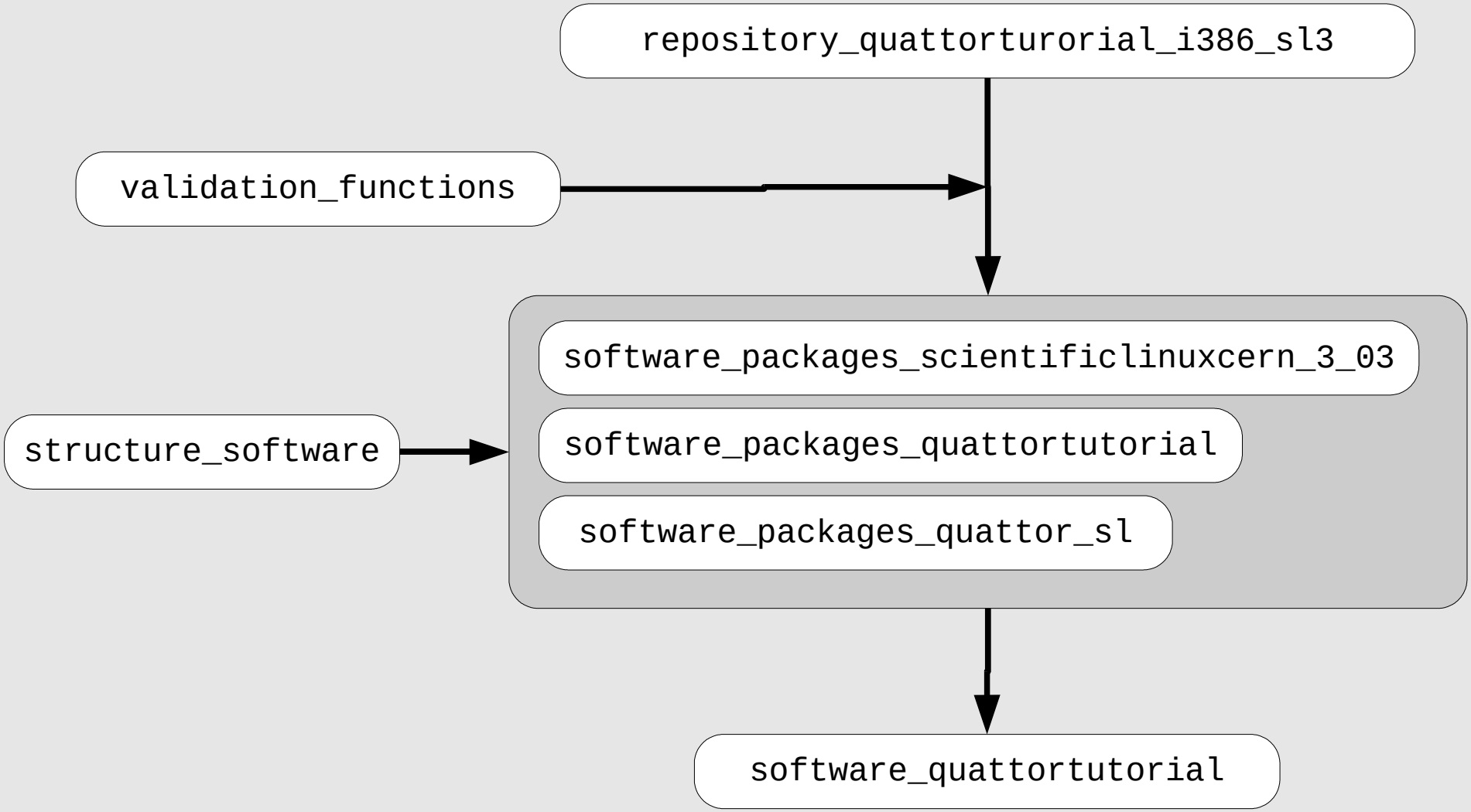
Template:

- `pro_declaration_functions_general`
- `pro_declaration_functions_network`
- `pro_declaration_functions_filesystem`

Function	Description
`resolve_pkg_rep`	Automatically fill "repository" field for package list
`purge_rep_list`	Remove unused repositories
`pkg_add`	Add package to the list
`pkg_del`	Remove package from list
`pkg_repl`	Replace package in the list
`pkg_ronly`	Replace package in the list only if present

Site Specific Templates

Templates for Software Management (SWRep/SPMA)



Templates for Software Management

How to create a repository template:

```
swrep-client template i386_sl3 >
repository_quattortutorial_i386_sl3.tpl
```

Please mind:

- Do not use the example template
- Do not edit this template by hand
- Template name is derived from repository name
  (see `swrep-server.conf`)
- Filename must be the same than template name

Templates for Software Management

Example of software packages organization:

`pro_software_packages_scientificlinuxcern_3_03`	SLC303 List of Packages (including updates)
`pro_software_packages_quattortutorial`	Extra packages and my own packages
`pro_software_packages_quattor_sl`	Quattor client packages + configuration components

Create your own list of packages by installing a target machine and then:

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

Templates for Software Management

Putting it all together: `pro_software_quattortutorial`

```
[...]

# SL(C)3 packages
include pro_software_packages_scientificlinuxcern_3_03;

# Quattor packages for clients
include pro_software_packages_quattor_sl;

# Extra packages for this cluster
include pro_software_packages_quattortutorial;

# create the repository
"/software/repositories/0" =
    create("repository_quattortutorial_i386_sl3");

# standard stuff
# resolve repository and purge not used entries
"/software/packages"     =
    resolve_pkg_rep(value("/software/repositories"));
"/software/repositories" =
    purge_rep_list(value("/software/packages"));
```

Templates for Hardware Management

- Hardware information is used for:
  - Configuration information validation (very little now)
  - Auditing information (queries through SQL server module)
  - Future ELFms modules (example: LEAF)
- Mandatory Fields:
  - CPU: Could be optional in quattor 1.0.1
  - RAM: Could be optional in quattor 1.0.1
  - Cards/NIC: Hardware address mandatory for AII
  - Hard Disks: Filesystem partitions could depend on number and size of hard disks.

Templates for Hardware Management

- Example of Hardware Templates:
  - CPU: `pro_hardware_cpu_Intel_Pentium_4_2600`
  - RAM: `pro_hardware_ram_1024`
  - NIC: `pro_hardware_card_nic_intel_e100`
  - Hard Disk: `pro_hardware_harddisk_STD_80`
- They are included in:
  - Node: `pro_hardware_generic_quattortutorial`

Templates for System Management

Example: How to organize configuration information

- Three configuration levels
- Each level inherits previous settings
- Each level overwrites previous settings

Level 1.- Global site configuration information

Template: `pro_system_base`

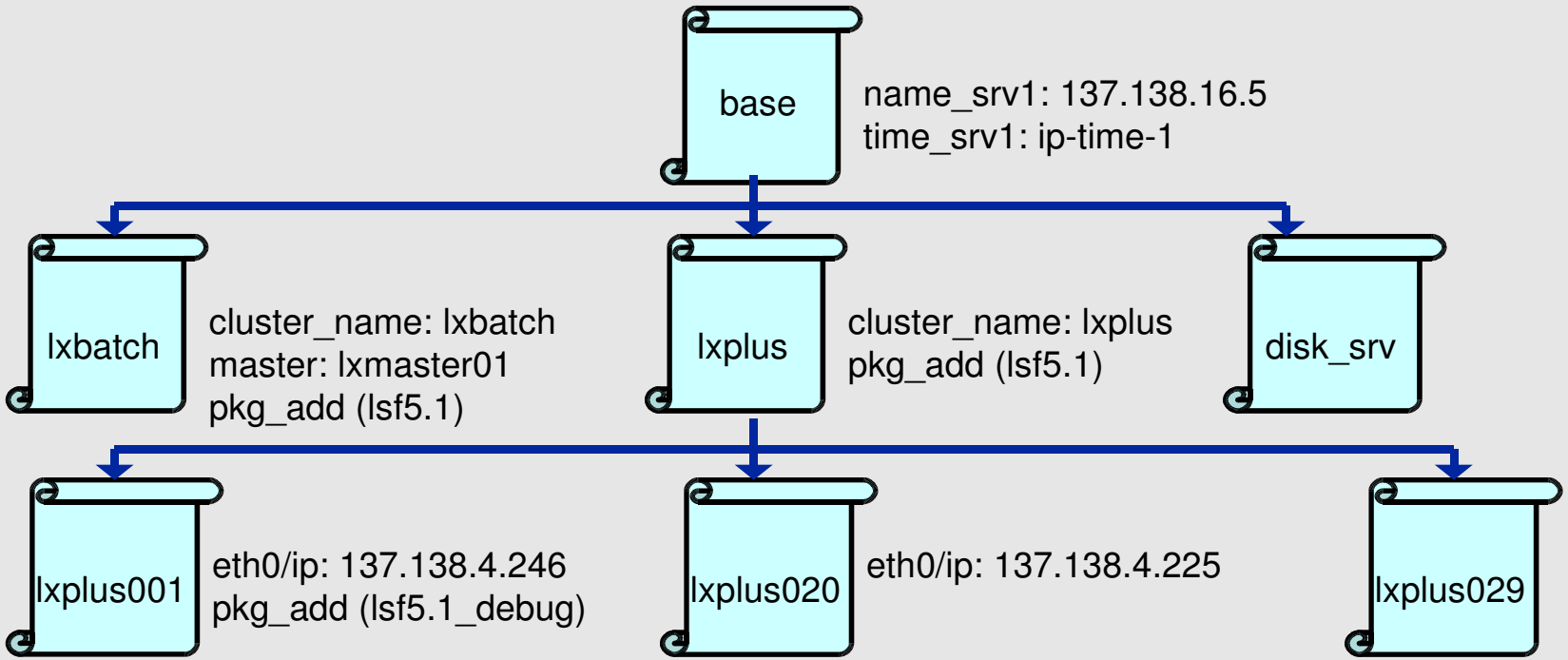
Level 2.- Per cluster configuration information

Template: `pro_system_quattortutorial`

Level 3.- Node specific configuration

Template: `profile_lxb0xxx`

Templates for System Management



Templates for System Management

Site configuration information (template: `pro_system_base`**)**

- **Global network settings**
  - domain name
  - name servers
  - time servers
- **Core configuration components**
  - spma
  - grub
  - ntpd
  - interactive limits
  - cron
  - accounts
- **Default NIC settings**
  - netmask
  - broadcast address
  - gateway
- **General AII settings**
  - OS installation Server
  - CDB server
  - others

Templates for System Management

Cluster configuration information (template: `pro_system_foo`**)**

- **Cluster information**
  - name
  - type
  - state
  - ...
- **Filesystem partitions**
  - size
  - type
  - mount points
- **Software package list**
- **Kernel version**
- **Additional components list**
  - access control
  - logrotate
  - authconfig
  - iptables
  - lmsensors
  - ...

UAM

quattor

Configuration Components Templates

Configuration Components Templates

Structure shared by all components:

- `active`: is the component active?
- `dispatch`: run automatically on changes (via `cdispd`)?
- `register_change`: list of subtree entries that affect the component
- `dependencies`: dependencies of other components
  - `pre`: list of components should run before
  - `post`: list of components should run after
- Specific component entries.

Configuration Components Templates

Component Declaration:

Example: `pro_declaration_component_accounts`

```
define type structure_component_accounts = {
        include structure_component
        "rootpwd"    ? boolean
        "shadowpwd"  ? boolean
        "users"      : structure_userinfo{}
        "groups"     : structure_groupinfo{}
};

type "/software/components/accounts"
     = structure_component_accounts;
```

Configuration Components Templates

Component Default Settings:

Example: `pro_software_component_accounts`

```
# standard component settings
"/software/components/accounts/active" =
                                        default( true );
"/software/components/accounts/dispatch" =
                                        default( true );
"/software/components/accounts/dependencies/pre" =
                                        default( list("spma") );


# component specific settings:
include pro_software_component_accounts_sysgroups;
include pro_software_component_accounts_sysusers;
```

UA

quattor

Client Node Templates

Creating a Client Node Template

Template: `profile_hostname`

```
object template profile_lxb0xxx;

  # structure of the profile
  include pro_declaration_profile_base;

  # hardware information
  include pro_hardware_generic_quattortutorial;
  "/hardware/cards/nic/eth0/hwaddr"                = "xx:xx:xx:xx:xx:xx";

  # network settings
  "/system/network/hostname"                       = "lxb0xxx";
  "/system/network/interfaces/eth0/ip"             = "xx.xx.xx.xx";

  # configuration information
  include pro_system_base;
  include pro_system_foo;

  # other host specific configuration information (optional)
  # "/hardware/serialnumber" = "0123456789";
  # "/system/kernel/version" = "2.4.25+custom+bar1";
```

Pan User Conventions Guide

Goal: *To describe a set of generic structures to store configuration information of a computer system.*

Notes:

- *Pan User Conventions* (RFC document finished)
- *Pan User Conventions* and *Pan Core Set* being synchronized