-WP4 Workshop- Draft proposal for a Fabric Global Schema

Maite Barroso

20/06/2002

Maite.Barroso.Lopez@cern.ch

Task Description

TASK:

- Definition of a common structure for fabric configuration information.

  This task has to propose a common scheme for all fabric configuration information to be stored in the Configuration Database, and has therefore to closely work together with all WP4 tasks.

Motivation

- Allows to share data between components/programs
- Decouples data from implementation (e.g. the filesystem configuration should not change regardless on if we use KickStart or LCFG or anything else to create the filesystems), this eases migration between different tools/components
- Allows to have system independent configuration representation (e.g. same filesystem description, validation, policies... for Linux and Solaris)
- Take advantage of other work like DMTF -> interoperability
- Others do the same way (e.g. hardware description in Windows registry)

Procedure

- Work together with CERN and Testbed system administrators to understand their configuration needs.
- Take a CERN lxplus standard node and extract its configuration information.
- Structure it (XML).
- Have a look to the DMTF standard, and compare it with the previous structure.
- Implement the global schema using the High level Description Language (HLDL) from the Configuration Mgt. Task.

XML vs. HLD schema presentation

- Present XML schema
- Present associated HLD schema, focusing in:
  - Template definitions to structure the data according to the defined schema
  - Data Validation
  - HLD Language capabilities

General XML Profile Structure

```
  <?xml version="1.0" encoding="utf-8" ?>
- <nlist name="profile" >
   - + <nlist name="system">
   - + <nlist name="hardware">
   - + <nlist name="sw">
- </nlist>
```

XML /hardware branch

```
- <nlist name="hardware">
    - + <nlist name="tape" type="record">
    - + <nlist name="dvd" type="record">
    - + <nlist name="cdrom" type="record">
    - + <nlist name="keyboard" type="record">
    - + <nlist name="mouse" type="record">
    - + <list name="CPUs">
    - + <list name="harddisks">
    - + <nlist name="network">
    - + <nlist name="memory">
- </nlist>
```

HLD /hardware branch

```
define type HARDWARE = {
        CPUs : CPUS
        harddisks : HARDDISKS
        tape : DEVICE
        dvd : DEVICE
        cdrom : DEVICE
        keyboard : DEVICE
        mouse : DEVICE
        network : NETWORK_I
        memory: MEMORY
};
type "/hardware" = HARDWARE;
```

XML /hardware/harddisks branch

```
- <list name="harddisks">
- - <nlist derivation="node_profile" type="record">
    - <string name="vendor">IBM</string>
    - <string name="model">DTLA-307045</string>
    - <string name="serial_number">1234560</string>
    - <string name="sys_name">hda</string>
    - <string name="interface_type">IDE</string>
    - <long name="size">43979</long>
    - <long name="cylinders">5606</long>
    - <long name="heads">255</long>
    - <long name="sectors">63</long>
- </nlist>
- + <nlist type="record">
- </list>
```

HLD /hardware/harddisks branch

❖ Validation:

```
define type HARDDISK = {
                sys_name        : string
                interface_type  : string
                vendor          : string
                model           : string
                serial_number   : string
                size            : long
                cylinders       : long
                heads           : long
                sectors         : long
};
define type HARDDISKS = HARDDISK[];
```

❖ Filling:

```
structure template disk_ibm_dtla_307045;
"interface_type" = "IDE";
"vendor" = "IBM";
"model" = "DTLA-307045";
"serial_number" = "1234560";
"size" = 43979;
"cylinders" = 5606;
"heads" = 255;
"sectors" = 63;

"/hardware/harddisks/0/sys_name"="hda";
```

XML /system branch

```
- <nlist name="system" derivation="testbed001">
    - <string name="hostname">testbed001</string>
    - <string name="architecture">i386_rh62</string>
    - <string name="localdomain">cern.ch</string>
    - + <nlist name="pam" type="table">
    - + <nlist name="nsswitch" type="record">
    - + <list name="inetd">
    - + <nlist name="partitions" type="table">
    - + <list name="filesystems">
    - + <list name="network_interfaces">
    - + <nlist name="dns" type="record">
    - + <list name="services">
    - + <nlist name="time">
- </nlist>
```

HLD /system branch

```
- define type SYSTEM = {
            partitions : PARTITIONS
            filesystems : FILESYSTEMS
            network_interfaces : NETWORKINTERFACES
            dns : DNS
            services : SERVICES
            time : TIME
            pam: PAM_CONFIGURATION
            nsswitch : NSSWITCH
            inetd : INETD
            localdomain : string
            architecture : string
            hostname : string
};
type "/system" = SYSTEM;
```

XML /system/partitions branch

```
- <nlist name="partitions" type="table">
    - - <nlist name="hda1" type="record">
        - <long name="size">38859</long>
        - <string name="partition_type">primary</string>
        - <string name="id">linux</string>
    - </nlist>
    - - <nlist name="hda2" type="record">
        - <long name="size">0</long>
        - <string name="partition_type">extended</string>
        - <string name="id">extended</string>
    - </nlist>
    - + <nlist name="hda5" type="record">
    - + <nlist name="hda6" type="record">
    - + <nlist name="hda7" type="record">
    - + <nlist name="hda8" type="record">
    - + <nlist name="hdc1" type="record">
    - + <nlist name="hdc2" type="record">
- </nlist>
```

HLD /system/partitions branch

- Validation:

```
define type PARTITION = {
                size           : long
                partition_type: string
                id             : string
};
define type PARTITIONS = PARTITION{};
```

- Filling:

```
"/system/partitions" = nlist(
     "hda1", nlist("size", 38859, "partition_type", "primary", "id", "linux"),
     "hda2", nlist("size", 0, "partition_type", "extended", "id", "extended"));
```

XML /system/nsswitch branch

```
- <nlist name="nsswitch" type="record">
    - - <list name="aliases">
          - <string>files</string>
          - <string>nisplus</string>
    - </list>
    - + <list name="services">
    - + <list name="passwd">
    - + <list name="shadow">
    - + <list name="group">
    - + <list name="hosts">
    - - <list name="bootparams">
          - <string>nisplus</string>
          - <string>[NOTFOUND=return]</string>
          - <string>files</string>    </list>
    - + <list name="ethers">
    - + <list name="netmasks">
    - + <list name="networks">
    - + <list name="protocols">
    - + <list name="rpc">
    - + <list name="netgroup">
    - + <list name="publickey">
    - + <list name="automount">
- </nlist>
```

HLD /system/nsswitch branch (validation)

- define type NSSWITCH_ELEMENT = string with

 match(self, '^(nisplus|nis\+|nis|yp|dns|files|db|compat|hesiod)$') ||

 match(self, '^\[!?(success|notfound|unavail|tryagain)

=(return|continue)\]$');

- define type NSSWITCH_SPEC = NSSWITCH_ELEMENT[1..] with

  match(self[0], '^(nisplus|nis+|nis|yp|dns|files|db|compat|hesiod)$');

- define type NSSWITCH = {

```
  passwd       : NSSWITCH_SPEC
  shadow       : NSSWITCH_SPEC
  group        : NSSWITCH_SPEC
  hosts        : NSSWITCH_SPEC
  bootparams   : NSSWITCH_SPEC
  ethers       : NSSWITCH_SPEC
  netmasks     : NSSWITCH_SPEC
  networks     : NSSWITCH_SPEC
  protocols    : NSSWITCH_SPEC
  rpc          : NSSWITCH_SPEC
  services     : NSSWITCH_SPEC
  netgroup     : NSSWITCH_SPEC
  publickey    : NSSWITCH_SPEC
  automount    : NSSWITCH_SPEC
  aliases      : NSSWITCH_SPEC
};
```

HLD /system/nsswitch branch (filling)

```
"/system/nsswitch/passwd" = list("files", "nisplus", "nis");
"/system/nsswitch/shadow" = list("files", "nisplus", "nis");
"/system/nsswitch/group" = list("files", "nisplus", "nis");
"/system/nsswitch/hosts" = list("files", "nisplus", "nis", "dns");
"/system/nsswitch/bootparams" = list("nisplus", "[NOTFOUND=return]", "files");
"/system/nsswitch/ethers" = list("files");
"/system/nsswitch/netmasks" = list("files");
"/system/nsswitch/networks" = list("files");
"/system/nsswitch/protocols" = list("files");
"/system/nsswitch/rpc" = list("files");
"/system/nsswitch/services" = list("files");
"/system/nsswitch/netgroup" = list("nisplus");
"/system/nsswitch/publickey" = list("nisplus");
"/system/nsswitch/automount" = list("files", "nisplus");
"/system/nsswitch/aliases" = list("files", "nisplus");
```

XML /sw branch

```
- <nlist name="sw">
    - + <nlist name="packages" type="table">
    - + <nlist name="monitoring" type="record">
- </nlist>
```

HLD /sw branch

```
define type SW = {
        packages : RPM_PACKAGE{}
        monitoring : MSA
};
type "/sw" = SW;
```

XML /sw/packages branch

```
- <nlist name="sw" derivation="rpm_profile">
    - - <nlist name="packages" type="table">
        - - <nlist name="edg_lcas" type="record">
              <string name="version">1.0.0-1</string>
              <string name="architecture">i386</string>
            - <nlist name="repositories" type="table">
                -   <nlist name="testbed" type="record">
                -   <string name="url">http://datagrid.in2p3.fr/</string>
                -   <string
                  name="path">/distribution/datagrid/wp4/gridification/RPMS</string>
                -</nlist>
              </nlist>
          </nlist>
        - - <nlist name="fabric_monitoring" type="record">
          </nlist>
      </nlist>
```

HLD /sw/packages branch (validation)

```
define type RPM_REPOSITORY= {
        url : string
        path : string
};

define type RPM_PACKAGE = {
        version : string
        architecture : string
        flags ? string[]
        repositories: RPM_REPOSITORY{}
};
```

HLD /sw/packages branch (filling)

```
# RPM edg-lcas-1.0.0-1.i386.rpm
structure template edg_lcas_rep;
"url" = "http://datagrid.in2p3.fr/";
"path" = "/distribution/datagrid/wp4/gridification/RPMS";

structure template edg_lcas;
"version" = "1.0.0-1";
"architecture" = "1386";
"repositories" = nlist(
                "testbed", create("edg_lcas_rep"));

/sw/packages" = nlist("edg_lcas", create("edg_lcas"));
```

Example 1: Add a new package to the list of RPMs to be installed in a node (1)

✓ **Repository structure maintained by the "product owners":**

```
/sw/known_repositories/Arep/url
                           /owner
                           /extras
                           /directories/dir_name_X/path
                                                  /platform
                                                  /packages/pck_a/name
                                                                 /version
                                                                 /architecture
                                       /dir_name_Y /path
                                                  /platform
                                                  /packages/pck_b/name
                                                                 /version
                                                                 /architecture
```

Example 1: Add a new package to the list of RPMs to be installed in a node (2)

✓ RPM information to be included in the node profile:

```
/sw/used_repositories/0/rep_name_A =
                       /1/rep_name_B =
/sw/packages/package_name/version =
                          /arch =
                          /flags =
                          / repositories/rep1/url =
                                                /path =
                                         /rep2/url
                                                /path
```

Different choices to fill this structure,

depending on the desired level of validation

Example 1: Add a new package to the list of RPMs to be installed in a node (3)

```
/sw/used_repositories/0/rep_name_A =
                       /1/rep_name_B =
/sw/packages/package_name/version =
                          /arch =
                          /flags =
                          / repositories/rep1/url =
                                             /path =
                                        /rep2/url  =
                                             /path =
```

This will be filled by the user

3 options

1	2	3
Leave empty Don't validate	Validate that the RPMs exist In the used_repositories	Write a function to fill automatically the optional part taking data from the repository structure

Example 2: Add a new node to an existing cluster

✓ **Configuration for the node:**

```
# testbed001 node profile
object template testbed001;

# include standard node profile
include node_profile;

# modify specific node information
"/system/network_interfaces/0/ip_address" = "137.138.30.48";
"/system/hostname" = "testbed001";
```

Example 3: Add a new disk to an existing node

1. Check that the template for the type of disk to be added already exist.
2. If it does not exist, create the new template :

```
structure template disk_ibm_dtla_307045;
include disk;
"interface_type" = "IDE";
"vendor" = "IBM";
"model" = "DTLA-307045";
"serial_number" = "1234560";
"size" = 43979;
"cylinders" = 5606;
"heads" = 255;
"sectors" = 63;
```

3. If it exist, e.g. disk_ibm_dtla_307045, modify the machine profile to add it:

```
object template testbed001;
"/hardware/harddisks" = merge( value("/hardware/harddisks"),
                        create("disk_ibm_dtla_307045"));
"/hardware/harddisks/1/sys_name"="hdc";
```

Example 4: X11 configuration

X11 configuration includes ~100 compulsory config parameters + ~50 optional ones, with many different types.
Different ways of including it into the global schema:

1) Total abstraction : Incorporate the whole X configuration information into the schema:

“/system/X/files/font_path” = …

“/rgb_path” = …

- Complex
- [illegible] validity, sysadmins only change <10%
- Validation

2) No abstraction: Just add a “reference” to an external file containing the desired configuration, which will not be included inline in the HLD, it is taken from the chosen server

Type “/system/X” = fetch;

“/system/X” = http://cern.ch/standard_cfg/X/XF86Config-4

- Simple
- No validation

3) Mixed solution:

Reference to external file with standard configuration

A few parameters representing the most often changed configuration information

- “/system/X” = http://cern.ch/standard_cfg/X/XF86Config-4
- “/system/X/files/font_path” =“tytytyt”;
- Flexible
- Validation where needed

DMTF

Common Information Model (CIM) Standards:

- ✓ It's a model for describing overall management information in a network enterprise environment.
- ✓ It is an approach to the management of systems and networks that applies the basic structuring and conceptualization techniques of the object-oriented paradigm.
- ✓ It is divided into a *Core model*, *Common model* and *extended schemas*
- ✓ Main components:
  - ✓ *Schema* is a group of classes with a single owner. Schemas are used for administration and class naming. Class names must be unique within their owning schemas.
  - ✓ *Class* is a collection of instances that support the same type: that is, the same properties and methods.
  - ✓ A *Property* is a value used to characterize instances of a Class.
  - ✓ *Method*
  - ✓ *Trigger* is a recognition of a state change (such as create, delete, update, or access) of a Class instance
  - ✓ *Indication, Association, References, Qualifiers,*

DMTF

CIM Device specification:

- ✓ Very wide device concept: It includes Cooling, Power, & Battery devices…
- ✓ Mixture of what we classify as monitoring and configuration information all over the schema:
  - ✓ Memory:
    - ✓ ErrorInfo: List of more recent errors
    - ✓ CorrectableError: Boolean indicating that the most recent error was " correctable.

DMTF EXAMPLES: CIM device schema

```
define type CONTROLLER = {
        ProtocolSupported : string
        MaxNumberControlled : long
};
define type SCSI_CONTROLLER = {
        include CONTROLLER
        MaxDataWidth : long
        MaxTransferRate : long
};
define type VIDEO_CONTROLLER = {
        include CONTROLLER
        ProtocolSupported : string
        VideoMemoryType : string
        NumberOfVideoPages : long
        MaxMemorySupported : long
        CurrentBitsPerPixel : long
        CurrentHorizontalResolution : long
        CurrentVerticalResolution : long
        MaxRefreshRate : long
        MinRefreshRate : long
        CurrentRefreshRate : long
        CurrentScanMode : string
        CurrentNumberOfRows : long
        CurrentNumberOfColumns : long
        CurrentNumberOfColors : long
};
```