# Node View Profile Specification (LLD-XML) version 2.0

Piotr Poznański

# Overview (1)

- Vehicle to transport configuration information from the CDB to client nodes,
- It is an XML based language,
- It is generated out of the HLD by the Pan language compiler (pan),
- Currently pan supports 5 formats – it is desirable to decrease this number,

# Overview (2)

- Two candidates:
  - "Node View Profile Specification" (Syntax 1),
  - New profile syntax (Syntax 2).
- The possibility and the way of validation is the most important issue concerning the choice. It is strictly related to complexity of LLD-XML language syntax.

# LLD-XML Example – HLD

```
object template pnlist;

"/device" = "/dev/hda";
"/partitions/root" = nlist ("size", 1000,
                            "mount", "/");
"/partitions/tmp"  = nlist ("size", 250,
                            "mount", "/tmp");

# PATH                        VALUE
# /device                     = "/dev/hda";
# /partitions/root/size       = 1000,
# /partitions/root/mount      = "/");
# /partitions/tmp/size        = 250,
# /partitions/tmp/mount       = "/tmp");
```

# LLD-XML Example – Syntax 1

```
<profile cfg:type="resource">
  <device cfg:type="string">/dev/hda</device>
  <partitions cfg:type="resource">
    <root cfg:type="resource">
      <size cfg:type="long">1000</size>
      <mount cfg:type="string">/</mount>
    </root>
    <tmp cfg:type="resource">
      <size cfg:type="long">250</size>
      <mount cfg:type="string">/tmp</mount>
    </tmp>
  </partitions>
</profile>
```

# LLD-XML Example – Syntax 2

```
<?xml version="1.0" encoding="utf-8"?>
<nlist name="profile">
  <string name="device">/dev/hda</string>
  <nlist name="partitions">
    <nlist name="root">
      <long name="size">1000</long>
      <string name="mount">/</string>
    </nlist>
    <nlist name="tmp">
      <long name="size">250</long>
      <string name="mount">/tmp</string>
    </nlist>
  </nlist>
</nlist>
```

# Validation

- We distinguish three levels:
  1. XML well formed,
  2. Parsed and accepted by (CCM-)fetch,
  3. Compliant with the data schema.
- Validation performed using XML Schema (as much more powerful than DTD).

# Syntax 1, Pros and Cons (1)

- All three levels of validation possible (+)
- Level 2 and 3 cannot be separated (-)
- XML schema (and the language syntax) is not finite and depends on data schema (-)
- XML schema has to be regenerated every time data schema is changed (-)
- XML schema is not compact and can grow with the size of data schema (-)

# Syntax 1, Pros and Cons (2)

- Possible problems of synchronising XML schema with contents of XML-LLD (-),
- Overuse of XML name spaces. Names of LLD elements populate XML element namespace (-), (`<foo>1</foo>` vs. `<foo>a</foo>` vs. `<foo><bar/></foo>`),
- XML constructs mixed with data contents - HLD elements names has to be restricted to correct XML element names (-) (e.g. "**cfg:foo**").

# Syntax 2, Pros and Cons (1)

- Level 3 validation is not possible with XML schema (-), but perhaps it is not required or may be handled outside of XML schema,
- XML schema is finite and does not change with changes of data schema (+),
- XML schema is compact (+),
- Only one namespace is needed (+)
- It will not impose any constraints on data contents and HLDL (+)