CPS Model Driven PoC

Tony Finnerty; Ciaran Johnston

Date, June 25 2020
Heterogeneous networks (that evolve!)

- Multiple vendors providing equipment / xNF
- Several telecom networks one ONAP platform
- Evolution of xNF and network standards
- ONAP applications and use cases competing for access to the same CM data
Blackboard pattern (for many consumers)
YANG (Netconf modelling language)

- Industry momentum towards YANG
- IETF RFC, widely supported
- Superset of most language capabilities

- Primary input and native language of the CPS
  - Other languages would need to be translated before being deployed
ONAP Design time model handling
Model driven safe access to data

- Model defines constraints to the data
- CPS enforces those constraints during access

- Developer efficiency
- Consistency of approach
- Lower maintenance costs; higher quality applications
Target

- **[Base scope] Read/write persisted Configuration Management data:**
  - defined by xNF, published as YANG
  - Models deployed @ runtime with no ONAP platform impacts or LCM events
  - Show the benefits in terms of constraint validation, access and upgrade

- **[Stretch target] Adapt behaviour of CPS on read/write based on information in the model**
  - Change notification emissions
  - Interaction with temporal data store, synergise with State Management PoC
  - Volatile xNF data read-through (e.g. state data)

- **Seed code for stand-alone CPS project in ONAP**
Proof points for base scope

• Demonstrate Create/read operations using YANG fragments against a CPS backed by very simple schema / schema-less repository

• Demonstrate ability to deploy / upgrade YANG fragments at runtime

• Demonstrate CPS behavior driven by YANG model

• Provide architecture vision and roadmap for a target architecture, supported use cases, non-functional requirements towards an ONAP Project
• Models provided by SDC (and controllers)
• Applications query models to take advantage of safe data access
• OLTP is primary/first focus
• State (volatile) pass-thru for convenience
• Other DB technologies: loosely coupled via model driven change notifications
Main interfaces and modules

- Sample deployment view
- Core functionality and REST interface are separate modules
- DBMS access via a Service Provider Interface
- Model handling will depend on interfaces and type safety – does not need to be in POD
- xNF State reader is for information only, not likely to be part of PoC
- DBMS is in own POD
Example of generic schema for relational DB

```plaintext
<table>
<thead>
<tr>
<th>data_relation_mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_fdn_id</td>
</tr>
<tr>
<td>to_fdn_id</td>
</tr>
<tr>
<td>from_mo_type_id</td>
</tr>
<tr>
<td>to_mo_type_id</td>
</tr>
<tr>
<td>rel_type_id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data_attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdn_id</td>
</tr>
<tr>
<td>fdn</td>
</tr>
<tr>
<td>root_fdn_id</td>
</tr>
<tr>
<td>mo_type_id</td>
</tr>
<tr>
<td>root_fdn</td>
</tr>
<tr>
<td>attributes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>relation_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rel_type_id</td>
</tr>
<tr>
<td>rel_type_name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mo_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>mo_type_id</td>
</tr>
<tr>
<td>mo_type_name</td>
</tr>
<tr>
<td>model_ref</td>
</tr>
</tbody>
</table>

jsonb*  actual type will depend on DB choice
Models at design and runtimes

- The runtime, at the bottom, involves parsing, mapping and storing – this is the CPS.
- Design time activity is once-off for each language to be supported.
- Parsing coupled with language (YANG).
- Mapping coupled with language and internal representation (Generic schema).
- Generic schema coupled with DBMS technology.
- CPS includes parsing and mapping.
- The java representation (in Yang Parser) may be used directly to avoid serialization.
- Not shown: CPS; SPI; Model repo; plug-ins; interfaces.
Q&A

• I’ve spoken enough – now it’s your turn!