YANG Mount

draft-clemm-netmod-mount

IETF 99 Prague

Alexander Clemm
Eric Voit
Reminder - what is “YANG-Mount”? + current status

• “The other mount” (actually, the original mount): Mounting Instances, not Schemas
  - Alias-Mount: Define an alternative path to instance information already elsewhere in the tree
  - Peer-Mount: Allow to access instance information authoritatively owned by a different server
    - Involves server infrastructure extension

• -06 posted, but draft has been kept virtually unchanged since -04

• Kept “dormant” as interest focused on Schema-Mount

• However, use cases for YANG-Mount remain valid (and may get more pressing)
  - Applications require visibility to data instantiated on other servers, or elsewhere on the same server
    - Instead of redundant configuration of parameters that need to be consistent, maintain link to authoritative copy
    - Provide context-awareness to state elsewhere in the network, e.g. to maintain network-wide service levels
    - Consolidated real-time network state (compare ODL MD-SAL)
    - Device to device, controller to device, device to controller, …
## Comparison YANG-Mount – Schema Mount

<table>
<thead>
<tr>
<th>YANG-Mount</th>
<th>Schema Mount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide visibility - create additional access path to instantiated data (local – alias and remote – peer)</td>
<td>Reuse existing definitions (schemas) to create new models</td>
</tr>
<tr>
<td>A server extension (also involves modeling construct)</td>
<td>A modeling construct</td>
</tr>
<tr>
<td>Analogy: soft link* (*with some caveats)</td>
<td>Analogy: grouping/uses (or augments)</td>
</tr>
<tr>
<td>Reference mount target has authoritative copy</td>
<td>Mount Point has authoritative copy</td>
</tr>
<tr>
<td>No validation of data at or by mountpoint; validation of data is responsibility of authoritative data owner</td>
<td>Validation of data at mount point</td>
</tr>
<tr>
<td>Mount point provides visibility to data already instantiated elsewhere (no redundant data)</td>
<td>Mountpoint instantiates data</td>
</tr>
<tr>
<td>The same target mounted in different mountpoints does not result in additional data instances</td>
<td>Same target schema mounted in different mountpoints results in separate unrelated data instances</td>
</tr>
</tbody>
</table>

**Commonality between YANG-Mount and Schema-Mount:**
- YANG mountpoint extension
- YANG extension introduced to define mountpoints
- Differences in terms of additional parameters (to identify target node and target system)
Refresher

• Super-impose new path structures on top of YANG data trees to access instantiated YANG data

• Commonality between YANG-Mount and Schema-Mount: YANG mountpoint extension
  YANG extension introduced to define mountpoints
  Differences in terms of additional parameters (to identify target node and target system)

• Allow YANG data nodes to link data in other (remote or local) subtree locations
  Insert (remote) subtrees under a mount point in a datastore
  Mount client: a YANG server that maintains the mounted “view”
  Mount server: the authoritative owner of the data
  For on-demand object access, mount server does not need to be aware of mount client
  Defines an alternative path to access data nodes
  Clients of the YANG server with mounted structure have visibility to it like “native” information

• Original draft emphasized remotability of data
  YANG Server allows its clients to access data that is conceptually federated across a network
  (Note: Peer-mount is also the basis for MD-SAL in Open Daylight, and is now proven/robust)

• However, mount points can also be defined for local data ✨ Alias Mount
Mount Concept – Peer Mount

Concept:
- Refer to data nodes / subtrees in remote datastores
- Remote data nodes visible as part of local data store
- Avoid need for data replication and orchestration
- Authority remains with original owner

Why:
- Federated datastore - treat network as a system
- “Borderless Agents”, “Network-as-a-System”
Notes

• Caching optimizations are possible (leverage YANG-Push)
• Circular mounting prohibited
• Focus on data nodes (not notifications)
• Original owner is authoritative
  Conditions/constraints/etc apply only there
• Mounted data intended as “read-only”
  Clients know the difference between data that is mounted vs data that is authoritatively owned
  This is about providing visibility, not about changing authority
  NMDA analogies apply
Next steps

• Editorial
  Editorial cleanup, relationship with schema-mount
  Investigate support for notifications referring to objects under their aliased name
  Consider extensions to accommodate dynamic configuration of mountpoints
  Review mountpoint construct for alignment between YANG-Mount and Schema-Mount

• Working group
  Solicit feedback – is there interest to move forward with this?
  Would fit well as a “next step” once schema mount is done

Thank you!
Backup
Usage example

- **rw controller-network**
  - **rw network-elements**
    - **rw network-element [element-id]**
      - **rw element-id**
      - **rw element-address**
        - **+-- M interfaces**

- **list network-element {**
  - key "element-id";
  - leaf element-id {
    - type element-ID;
  }
  - container element-address {
    - ...
  }
  - mnt:mountpoint "interfaces" {
    - mnt:target "./element-address";
    - mnt:subtree "/if:interfaces";
  }
}

---

**Module structure**

- YANG module defines YANG mount extensions + data model for mountpoint management

- YANG extensions:
  - **Mountpoint**: Defined under a containing data node (e.g. `container`, `list`)
  - **Target**: References data node that identifies remote server [peer-mount only]
  - **Subtree**: Defines root of remote subtree to be attached

---

**Mountpoint declaration**

```
<network-element>
  <element-id>NE1</element-id>
  <element-address> .... </element-address>
  <interfaces>
    <if:interface>
      <if:name>fastethernet-1/0</if:name>
      <if:type>ethernetCsmacd</if:type>
      <if:location>1/0</if:location>
    ... 
  </if:interface>
  ...
</network-element>
```

---

**Instance information**
Alias mount example

```
rw my-container
  +-- rw sub-container
  +-- M interfaces

...
Mountpoint management

rw mount-server-mgmt
  +-- rw mountpoints
    |  +-- rw mountpoint [mountpoint-id]
    |     +-- rw mountpoint-id  string
    |     +-- rw mount-target
    |     |     +--: (IP)
    |     |     |     +-- rw target-ip  yang:ip-address
    |     |     +--: (URI)
    |     |     |     +-- rw uri  yang:uri
    |     |     +--: (host-name)
    |     |     |     +-- rw hostname  yang:host
    |     |     +-- (node-ID)
    |     |     |     +-- rw node-info-ref  mnt:subtree-ref
    |     |     +-- (other)
    |     |     |     +-- rw opaque-target-id  string
    |     +-- rw subtree-ref  mnt:subtree-ref
    |     +-- ro mountpoint-origin enumeration
    |     +-- ro mount-status  mnt:mount-status
    |     +-- rw manual-mount?  empty
    |     +-- rw retry-timer?  uint16
    |     +-- rw number-of-retries?  uint8
  +-- rw global-mount-policies
    +-- rw manual-mount?  empty
    +-- rw retry-time?  uint16
    +-- rw number-of-retries?  uint8

• Mountpoints can be system-administered
• Applications&users are not exposed to this
• System administration can add bindings
  Update on-demand, periodic, on-change
• Not shown:
  Mount bindings - data update subscriptions
• Model needs updating to distinguish alias and peer mount