Mounting YANG-Defined Information from Remote Datastores

draft-clemm-netmod-mount-01.txt

Alexander Clemm, alex@cisco.com
Jan Medved, jmedved@cisco.com
Eric Voit, evoit@cisco.com
Purpose

• Allow YANG Datastores to reference information in remote datastores
• YANG Server (Netconf, RESTconf) allows applications to access data that is conceptually federated
• Applications/ use cases:
  – Incorporate information from remote systems into consolidated network view
  – Validation of parameter settings with cross-device dependencies
    • E.g global network policies, parameters, intent
    • Coordination/orchestration left to users/applications today
• Ask: Adopt as WG item
Datastore mount concept

- Allow data store to refer to remote data nodes / subtrees
- Remote data nodes conceptually treated as part of local data store
- Avoid need for redundant data modeling
- Avoid need for replication and orchestration
- Greater consistency
- Federated datastore - treat network as one
Datastore mount concept (contd.)

• Mount client:
  - Contains mount points at which to attach remote subtrees into data tree
  - Requests whose scope contains remote data are proxied/forwarded to remote system
  - Acts as application/client to the remote system

• Mount server
  - Authoritative owner of the data
  - May not be aware that mounting occurs
    (mount client is “just another application”)

• Notes
  - Caching optimizations possible, implementation dependent
  - Circular mounting prohibited
  - Primary usage: accessing/reading of data
    - Configuration is also possible; locking depends on ability to obtain mount server locks
  - Notifications and RPCs currently outside scope
Application example: Network controller

• Provide consolidated network view to applications north of controller without replicating information from controlled nodes
  – Mount information from devices and interfaces below nodes inventory
  – Allow to change containment hierarchy
    • E.g. place top level “system” information underneath list of nodes
    • Device and network abstractions complement one another in same data tree
  – No need for replicated of device models
  – Dynamic discovery and support of new device features
    • Controller not a bottle neck for the adoption of new feature
Network controller provided network view

- One datastore mounts many others
- Network controller presents network abstractions, mounts network element abstractions
- Avoid redundant inventory, maintain accurate network synchronization
- Single point of contact for external systems

Diagram:
- External systems
  - e.g. Netconf, RESTconf
- Network controller
  - Network element
  - Network element
  - Datastore scope
  - Network element
  - Network element
Mountpoint YANG module

YANG extensions:

Mountpoint
 Target: Reference data node that identifies remote server
 Subtree: Define root of remote subtree to be attached

```yarray
rw mount-server-mgmt
  +- rw mountpoints
    +- rw mountpoint [mountpoint-id]
      |   +- rw mountpoint-id string
      |   +- (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
RPCs for manual mount, unmount
```
Usage example

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

Mountpoint declaration

... 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";
  }
} ...

Instance information

<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>
    ...
  </interfaces>
</network-element>

Module structure
Questions?