Draft Proposal: Upgrades to the OpenDaylight PCMM API in Beryllium

Erich Arnold (erich.arnold@arris.com)
Engineering Fellow
Network Solutions CTO Team

Proposed Upgrades to ODL PCMM API for Beryllium – Part 1

- Problem: In the CCAP and Gate APIs the COPS operational state was previously contained in the "response" object that was inserted back into the restconf/config datastore object by the PCMM plugin when the COPS operation completed
  - The "response" object recorded the COPS operation state for the deployed object in the config datastore, but this is not conformant with the ODL RESTCONF config and operational datastore architecture
  - The config datastore "response" object was completely removed in Lithium SR1 due to conflicts with the distributed datastore feature thus leaving no COPS operational feedback in the PCMM API for the apps or devops
  - Note well that reverse maps and other in-memory cache objects maintained by the plugin for performance reasons that are based on objects created in the config datastore are not considered "state" as they are expendable and can be recreated from the config datastore at any time

ODL Li SR1 PCMM CCAP Config API: GET CCAP Config Request per Part 1

```
GET http://localhost:8181/restconf/config/packetcable:ccap/ccaps/CMTS-1
```

```
{"ccaps":[
         {"ccapId":"CMTS-1",
          ~~"response": "200 OK - CCAP connected: CMTS-1 @ 10.20.30.40:3918"~~,
          "amId": {
                  "am-tag": 51930,
                  "am-type": 1
          },
          "connection": {
                  "ipAddress": "10.20.30.40",
                  "port":3918
          },
          "subscriber-subnets": [
                  "2001:4978:030d:1000:0:0:0:0/52",
                  "44.137.0.0/16"
          ],
          "upstream-scns": [
                  "SCNA",
                  "extrm_up"
          ],
          "downstream-scns": [
                  "extrm_dn",
                  "ipvideo_dn",
                  "SCNC"
          ]}
    ]}
```

ODL Li SR1 PCMM QoS Gate Config API: Get Gate Config Request per Part 1

GET http://localhost:8181/restconf/**config**/packetcable:qos/apps/**testApp**/subs/**44.137.1.20**/gates/**ds-gate-1**

```
{"gates":[
         {"gateId":"ds-gate-1",
          ~~"response": "200 OK - created QoS gate for CMTS-1/testApp/
44.137.1.12/ds-gate-1 with COPS GateId e2090029"~~,
          "gate-spec":{
                  "dscp-tos-overwrite": 160,
                  "dscp-tos-mask": 255
          },
          "traffic-profile": {
                  "service-class-name": "ipvideo_dn"
          },
          "classifier":{
                  "dstPort": 4321,
                  "tos-mask": 224,
                  "srcIp": "10.10.10.1",
                  "srcPort": 1234,
                  "protocol": 0,
                  "dstIp": "44.137.1.12",
                  "tos-byte": 160
          }
         }
]}
```

Proposed Upgrades to ODL PCMM API for Beryllium – Part 1 (cont.)

- Solution: use the restconf/operational datastore to contain all COPS operational state information related to the CCAP and Gate restconf/config datastore objects
  - PUT to the restconf/config datastore creates the CCAP or Gate Object and immediately returns 200 OK unless the Yang model is not valid
  - The onDataChanged() thread validates the rest of the input JSON object and deploys the COPS Connection or Gate Object to the target CCAP
  - Once it completes, the result of the COPS operation is stored in the restconf/operational datastore using the same path as the restconf/config object
    - This is the read-only COPS operational "state" of the deployed restconf/config object
    - The operational state object always includes a "timestamp" object and an "error" list container that is either empty if the COPS operation succeeds or else identifies the COPS failures if the operation does not succeed
    - Additional object-specific state information is also included such as the COPS gateID and the target ccapID that are used to access the COPS object after creation
    - Note that data validation in the onDataChanged() thread can also create an restconf/operational object with an "error" list even if it fails the CCAP or Gate Request operation because it catches one or more semantic errors in the JSON object (such as no matching CCAP subnet for the subscriberID or a misspelled service class name)
  - GET to the restconf/operational datastore returns the object state containing the "error" list container where "error": [] is success.
  - DELETE to a restconf/config datastore object also removes the related restconf/operational datastore object

ODL Be PCMM CCAP Config API Upgrade : GET CCAP Operational Request per Part 1

```
GET http://localhost:8181/restconf/operational/packetcable:ccap/ccaps/CMTS-1
```

```
{"ccaps":[
         {"ccapId":"CMTS-1",
          "connection": {
              "ipAddress": "10.20.30.40",
              "port": 3918,
              "connected": true,
              "error": []
          },
          "timestamp": "2015-09-29 09:48:38,379"
         }
       ]}
```

Error response:

```
{"ccaps":[
         {"ccapId":"CMTS-1",
          "connection": {
              "ipAddress": "10.20.30.40",
              "port": 3918,
              "connected": false,
              "error": ["CMTS-1 COPS connection refused"]
         },
         "timestamp": "2015-09-29 09:48:38,379"
        }
      ]}
```

ODL Be PCMM QoS Gate Config API Upgrade : GET Gate Operational Request per Part 1

```
GET http://localhost:8181/restconf/operational/packetcable:qos/
apps/testApp/subs/44.137.1.20/gates/ds-gate-1
```

Success response:

```
{"gates":[
           {"gateId":"testApp/44.137.1.20/ds-gate-1",
            "ccapId": "CMTS-1",
            "cops-state": "active",
            "cops-gateId": "e2090029",
            "timestamp": "2015-09-29 09:48:38,379",
            "error": []
           }]
}
```

Error response reporting COPS Gate Request failure:

```
{"gates":[
           {"gateId":"testApp/44.137.1.20/ds-gate-1",
            "ccapId": "CMTS-1",
            "cops-state": "failed",
            "cops-gateId": null,
            "timestamp": "2015-09-29 09:48:38,379",
            "error": ["CMTS-1 reports SubscriberId 44.137.1.20 is offline"]
           }]
}
```

Error response reporting Gate Request semantic validation errors:

```
{"gates":[
           {"gateId":"testApp/44.137.1.20/ds-gate-1",
            "ccapId": null,
            "cops-state": null,
            "cops-gateId": null,
            "timestamp": "2015-09-29 09:48:38,379",
            "error": ["SubscriberId 10.10.10.10 not found in CCAP subnet map",
                      "Service Class Name scn_xyz not found in CCAP config"]
           }]
}
```

Proposed Upgrades to ODL PCMM API for Beryllium – Part 2

- Problem: CCAP COPS connections are continuously open even when there is no Gate Request activity initiated by the apps
  - A COPS "keep-alive" mode session is maintained 24/7 for the entire life of the restconf/config CCAP object regardless of Gate Request activity by the apps which may create performance issues in both ODL and the target CCAP
- Solution: Extend the CCAP config API to include an "idle-detect" object within the "connection" container that allows the COPS connection to be automatically closed when no Gate Request activity is detected for the specified period
  - "idle-detect" is the integer number of seconds that the COPS connection will remain open following the last Gate Request; zero seconds indicates that the COPS connection will remain open indefinitely (legacy behavior)
  - The " idle-detect " feature allows the CCAP COPS connection to be opened initially during CCAP object creation in normal "keep-alive" mode but only for the specified idle-detect period after which it will be automatically closed if no further Gate Requests or Gate Operations are received during the idle-detect period
    - Whenever new Gate Requests or Gate Operations are received while the COPS connection is closed, then the COPS connection is automatically opened in normal "keep-alive" mode for another idle-detect period during which all new Gate Requests or Operations are processed against the open COPS connection
    - Note that Operations on active COPS Gate Objects maintained in the operational datastore can also cause a COPS connection to be reopened to the target ccapId contained in the operational Gate Object
    - If a new Gate Request or Operation appears before the idle-detect period expires, then the idle-detect timer is reset to renew the idle-detect period again – so, yes a busy app can keep the COPS connection open indefinitely if it needs to without changing the "idle-detect" period in the CCAP "connection" config
    - However, if the idle-detect period expires, then the COPS connection is automatically closed pending new Gate Request or Operations arrivals
  - Depends on Part 1 operational data store

ODL Be PCMM CCAP Config API Upgrade: PUT CCAP Config Request per Part 2

```
PUT http://localhost:8181/restconf/config/packetcable:ccap/ccaps/CMTS-1

{"ccaps": [{
        "ccapId": "CMTS-1",
        "connection": {
            "ipAddress": "10.20.30.40",
            "port": 3918,
            "idle-detect": 120
        },
        "amId": {
            "am-tag": "0xcada",
            "am-type": "1"
        },
        "subscriber-subnets": [
            "44.137.0.0/16",
            "2001:4978:030d:1000:0:0:0:0/52"
        ],
        "downstream-scns": [
            "ipvideo_dn",
            "extrm_dn",
            "SCNC"
        ],
        "upstream-scns": [
            "SCNA",
            "extrm_up"
        ]
    }
  ]}
```

ODL Be PCMM CCAP Config API Upgrade : GET CCAP Operational Request per Part 2

```
GET http://localhost:8181/restconf/operational/packetcable:ccap/ccaps/CMTS-1
```

```
{"ccaps":[
         {"ccapId":"CMTS-1",
          "connection": {
              "ipAddress": "10.20.30.40",
              "port": 3918,
              "connected": true,
              "idle-detect": 120,
              "isIdle": true,
              "error": []
          },
          "timestamp": "2015-09-29 09:48:38,379"
         }
       ]}
```

Proposed Upgrades to ODL PCMM API for Beryllium – Part 3

- Problem: Gate configurations only support a single classifier
  - In particular dual-stacked CPE hosts often require both an IPv4 and an IPv6 classifier to properly identify the traffic carried by the service flow
- Solution: Modify the Gate config API to include a "classifiers" container that provides a list of one or more classifier objects
  - The "classifiers" list must include at least 1 and up to 4(?) classifiers due to COPS protocol limits
  - The classifier objects in the list are defined by any mix of the legacy classifier, extended classifier, and/or IPv6 classifier objects
  - The classifiers in the list have assumed classifier priority based on list position with the first classifier in the list being the highest priority classifier and proceding in descending order to the last classifier in the list
  - Thus, the classifiers in the list are assumed to be ordered from most-specific to least-specific allowing the last classifier to be a "catch-all" classifier if required

ODL Be PCMM QoS Gate Config API Upgrade :
PUT Gate Config Request per Part 3 with Single Classifier

```
PUT http://localhost:8181/restconf/config/packetcable:qos/
apps/testApp/subs/44.137.1.20/gates/ds-gate-1

{ "gates": [{
        "gateId": "ds-gate-1",
        "gate-spec": {"dscp-tos-overwrite": "0xa0",
                      "dscp-tos-mask": "0xff"},
        "traffic-profile": {"service-class-name": "ipvideo_dn"},
        "classifiers": [
             "ext-classifier": {
                 "srcIp": "10.10.10.0",
                 "srcIpMask": "255.255.255.0",
                 "dstIp": "44.137.1.12",
                 "dstIpMask": "255.255.255.255",
                 "tos-byte": "0xa0",
                 "tos-mask": "0xe0",
                 "protocol": "0",
                 "srcPort-begin": "1234",
                 "srcPort-end": "1235",
                 "dstPort-begin": "4321",
                 "dstPort-end": "4322"
             }
        ]
    }]
}
```

ODL Be PCMM QoS Gate Config API Upgrade :
PUT Gate Config Request per Part 3 with Multiple Classifiers

PUT http://localhost:8181/restconf/**config**/packetcable:qos/apps/**testApp**/subs/**44.137.1.20**/gates/**ds-gate-1**

```
{ "gates": [{
        "gateId": "ds-gate-1",
        "gate-spec": {"dscp-tos-overwrite": "0xa0",
                         "dscp-tos-mask": "0xff"},
        "traffic-profile": {"service-class-name": "ipvideo_dn"},
        "classifiers": [
            "ext-classifier": {
                    "srcIp": "10.10.10.0",
                    "srcIpMask": "255.255.255.0",
                    "dstIp": "44.137.1.12",
                    "dstIpMask": "255.255.255.255",
                    "tos-byte": "0xa0",
                    "tos-mask": "0xe0",
                    "protocol": "0",
                    "srcPort-begin": "1234",
                    "srcPort-end": "1235",
                    "dstPort-begin": "4321",
                    "dstPort-end": "4322"
             },
             "ipv6-classifier": {
                 "srcIp6": "2001:4978:030d:1000:0:0:0:0/64",
                 "dstIp6": "2001:4978:030d:1101:0:0:0:0/64",
                 "flow-label": "0",
                 "tc-low": "0xa0",
                 "tc-high": "0xa0",
                 "tc-mask": "0xe0",
                 "next-hdr": "256",
                 "srcPort-begin": "1234",
                 "srcPort-end": "1235",
                 "dstPort-begin": "4321",
                 "dstPort-end": "4321"
             }
         ]
     }]}
```

Proposed Upgrades to ODL PCMM API for Beryllium – Part 4

- Problem: CCAP and Gate configurations are static and cannot be operated on except by PUT/DELETE of the restconf/config object
- Solution: Use the POST restconf/operations path for RPC transactions that can be immediately performed by the plugin against existing CCAP or Gate restconf/config objects (and thus forces updates to their restconf/operational state objects)
  - CCAP RPC operations include commands to start or stop a COPS Connection that may be hung or to change the COPS Connection "idle-detect" seconds parameter which results in the update of state information in the restconf/operational objects
  - CCAP or Gate RPC operations are also used for on-demand polling of the current state of a CCAP COPS Connection or COPS Gate Object which results in the update of state information in the restconf/operational objects
  - Depends on Part 1 operational data store

ODL Be PCMM CCAP Config API Upgrade : Post CCAP Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:ccap/set-connections
```

Force COPS connection to stop:

```
{"input":
    { "ccapId": "/packetcable:ccap:ccaps/[packetcable:ccap:ccapId='CMTS-1']",
      "connection": {
          "connected": false
      }
    }
}
```

Response: 200 OK

```
{"output": {
    "ccaps":[
         {"ccapId":"CMTS-1",
          "connection": {
              "ipAddress": "10.20.30.40",
              "port": 3918,
              "idle-detect": 120,
              "isIdle": null,
              "connected": false,
              "error": []
          },
          "timestamp": "2015-09-29 09:48:38,379"
         }
      ],
     "response": "CMTS-1: CCAP operation complete",
     "timestamp": "2015-09-29 09:48:38,444"
}}
```

ODL Be PCMM CCAP Config API Upgrade : Post CCAP Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:ccap/set-connections
```

Change COPS idle detection parameters:

```
{"input":
    { "ccapId": "/packetcable:ccap:ccaps/[packetcable:ccap:ccapId='CMTS-1']",
      "connection": {
          "idle-detect": 60
      }
    }
}
```

Response: 200 OK

```
{"output": {
     "ccaps":[
         {"ccapId":"CMTS-1",
          "connection": {
              "ipAddress": "10.20.30.40",
              "port": 3918,
              "idle-detect": 60,
              "isIdle": true,
              "connected": true,
              "error": []
          },
          "timestamp": "2015-09-29 09:48:38,379"
         }
       ],
     "response": "CMTS-1: CCAP operation complete",
     "timestamp": "2015-09-29 09:48:38,444"
}}
```

ODL Be PCMM CCAP Config API Upgrade : Post CCAP Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:ccap/set-connections
```

Force COPS connection to start with idle detect turned off:

```
{"input":
    { "ccapId": "/packetcable:ccap:ccaps/[packetcable:ccap:ccapId='CMTS-1']",
      "connection": {
          "idle-detect": 0,
          "connected": true
      }
    }
}
```

```
Response: 200 OK
{"output": {
     "ccaps":[
         {"ccapId":"CMTS-1",
          "connection": {
              "ipAddress": "10.20.30.40",
              "port": 3918,
              "idle-detect": 0,
              "isIdle": false,
              "connected": true,
              "error": []
          },
          "timestamp": "2015-09-29 09:48:38,379"
         }
       ],
     "response": "CMTS-1: CCAP operation complete",
     "timestamp": "2015-09-29 09:48:38,444"

}}
```

ODL Be PCMM CCAP Config API Upgrade : Post CCAP Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:ccap/poll-connections

{"input":
    { "ccapId": "/packetcable:ccap:ccaps/[packetcable:ccap:ccapId='CMTS-1']"}
}

Response: 200 OK
{"output": {
     "ccaps":[
           {"ccapId":"CMTS-1",
            "connection": {
                "ipAddress": "10.20.30.40",
                "port": 3918,
                "idle-detect": 120,
                "isIdle": true,
                "connected": true,
                "error": []
            },
            "timestamp": "2015-09-29 09:48:38,379"
          }
       ],
     "response": "CMTS-1: CCAP poll complete",
     "timestamp": "2015-09-29 09:48:38,444"
}}

Or COPS Connection in Error State:
{"output": {
     "ccaps":[
           {"ccapId":"CMTS-1",
            "connection": {
                "ipAddress": "10.20.30.40",
                "port": 3918,
                "idle-detect": 120,
                "isIdle": null,
                "connected": false,
                "error": ["CMTS-1 COPS connection refused"]
           },
           "timestamp": "2015-09-29 09:48:38,379"
         }
      ],
     "response": "CMTS-1: CCAP poll complete",
     "timestamp": "2015-09-29 09:48:38,444"
}}
```

ODL Be PCMM Gate Config API Upgrade : Post Gate Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:qos/poll-gates
{"input":
    { "appId": "/packetcable:qos:apps/[packetcable:qos:appId='testApp']"
      "subId": "44.137.1.20",
      "gateId": "ds-gate-1"
    }
}

Response: 200 OK
{"output": {
     "gates":[
          {"gateId": "testApp/44.137.1.20/ds-gate-1",
           "ccapId": "CMTS-1",
           "cops-state": "active",
           "cops-gateId": "e2090029",
           "timestamp": "2015-09-29 09:48:38,379",
           "error": []
          }],
     "response": "testApp/44.137.1.20/ds-gate-1: gate poll complete",
     "timestamp": "2015-09-29 09:48:38,444"
}}

Or Active Gate in Error State:
{"output": {
     "gates":[
          {"gateId":"testApp/44.137.1.20/ds-gate-1",
           "ccapId": "CMTS-1",
           "cops-state": "offline",
           "cops-gateId": "e2090029",
           "timestamp": "2015-09-29 09:48:38,379",
           "error": ["CMTS-1 reports SubscriberId 44.137.1.20 is offline"]
          }],
     "response": "testApp/44.137.1.20/ds-gate-1: gate poll complete",
     "timestamp": "2015-09-29 09:48:38,444"
}}
```

ODL Be PCMM Gate Config API Upgrade : Post Gate Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:qos/poll-gates
{"input":
    { "appId": "/packetcable:qos:apps/[packetcable:qos:appId='testApp']"}
}

Response: 200 OK
{"output": {
     "gates":[],
     "response": "testApp/: gate subtree poll in progress",
     "timestamp": "2015-09-29 09:48:38,000"
}}

GET http://localhost:8181/restconf/operational/packetcable:qos/apps/testApp

Response: 200 OK
{"gates":[
         {"gateId":"testApp/44.137.1.20/ds-gate-1",
          "ccapId": "CMTS-1",
          "cops-state": "active",
          "cops-gateId": "e2090029",
          "timestamp": "2015-09-29 09:48:38,379",
          "error": []
         },
         {"gateId":"testApp/44.137.1.22/ds-gate-22",
          "ccapId": "CMTS-1",
          "cops-state": "offline",
          "cops-gateId": "a2090030",
          "timestamp": "2015-09-29 09:48:38,379",
          "error": ["CMTS-1 reports SubscriberId 44.137.1.22 is offline"]
         }]
}}
```

ODL Be PCMM Gate Config API Upgrade : Post Gate Operations Request per Part 4

```
POST http://localhost:8181/restconf/operations/packetcable:qos/poll-gates
{"input":
    { "appId": "/packetcable:qos:apps/[packetcable:qos:appId='testApp']",
      "subId": "44.137.1.20",}
}

Response: 200 OK
{"output": {
     "gates":[],
     "response": "testApp/44.137.1.20/: gate subtree poll in progress",
     "timestamp": "2015-09-29 09:48:38,000"
}}

GET http://localhost:8181/restconf/operational/packetcable:qos/apps/testApp/
subs/44.137.1.20

Response: 200 OK
{"gates":[
         {"gateId":"testApp/44.137.1.20/ds-gate-1",
          "ccapId": "CMTS-1",
          "cops-state": "active",
          "cops-gateId": "e2090029",
          "timestamp": "2015-09-29 09:48:38,379",
          "error": []
         }]
}}
```

Proposed Upgrades to ODL PCMM API for Beryllium – Part 5

- Problem: Gate configurations cannot be updated, they can only be created or deleted
- Solution: Extend the Gate Object config onDataChanged() thread processing to allow updates to the COPS Gate Objects
  - The initial PUT of the Gate Object creates and deploys the COPS object to the target COPS connection
  - A subsequent PUT of the same Gate Object could trigger a complex analysis of the delta between the original Gate Object and the updated Gate Object as both the original object and the updated object are passed to the onDataChanged() thread
  - More simply, if we require the replacement Gate Request Object to represent the final form of the new operational COPS gate (that is, not just the changes), then a seamless COPS Gate swap-out can be performed in a single COPS transaction by first setting the replacement COPS gate and then deleting the original COPS gate
    - Note well that the original dynamic Service Flow will continue to carry its classified traffic until it is deleted at which time the replacement dynamic Service Flow will catch the next packet with its new classifiers and no packets will be missed
    - Also note that this swap-out procedure allows any Gate parameters in the replacement Gate object to completely superscede the original Gate parameters (including SCN).
      - However, be aware that if you change the SCN in the replacement Gate Request, then this will be visible in the IPDR records that report SF counts with SCN tags