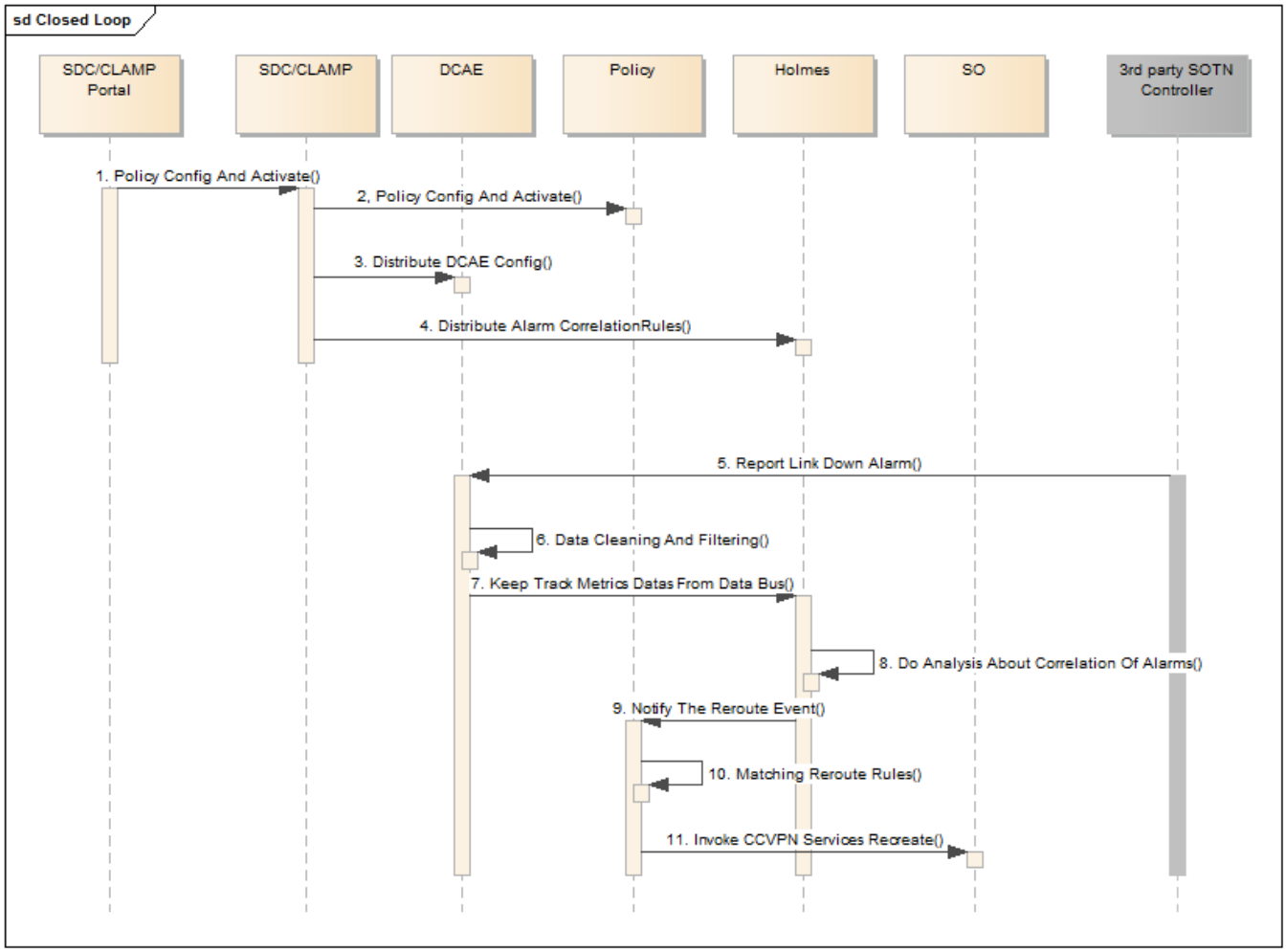
CCVPN Closed Loop

CCVPN Closed Loop Diagram



1. SDC/CLAMP Portal design and activate policy.
2. SDC/CLAMP config and activate the policy.
3. SDC/CLAMP distribute the DCAE config.
4. SDC/CLMAP distribute the alarm correlation rules to Holmes.
5. 3rd party SOTN controller report link down alarm to DCAE
6. DCAE will do data cleaning and filtering for the alarms
7. DCAEk keep track the datas.
8. Holmes do analysis for the alarms.
9. Holmes notify the reroute event.
10. Policy matching the reroute rules.
11. Policy call SO to delete the old services and create the new services. For the creation flow, a variable route will be recalculated.

CLAMP

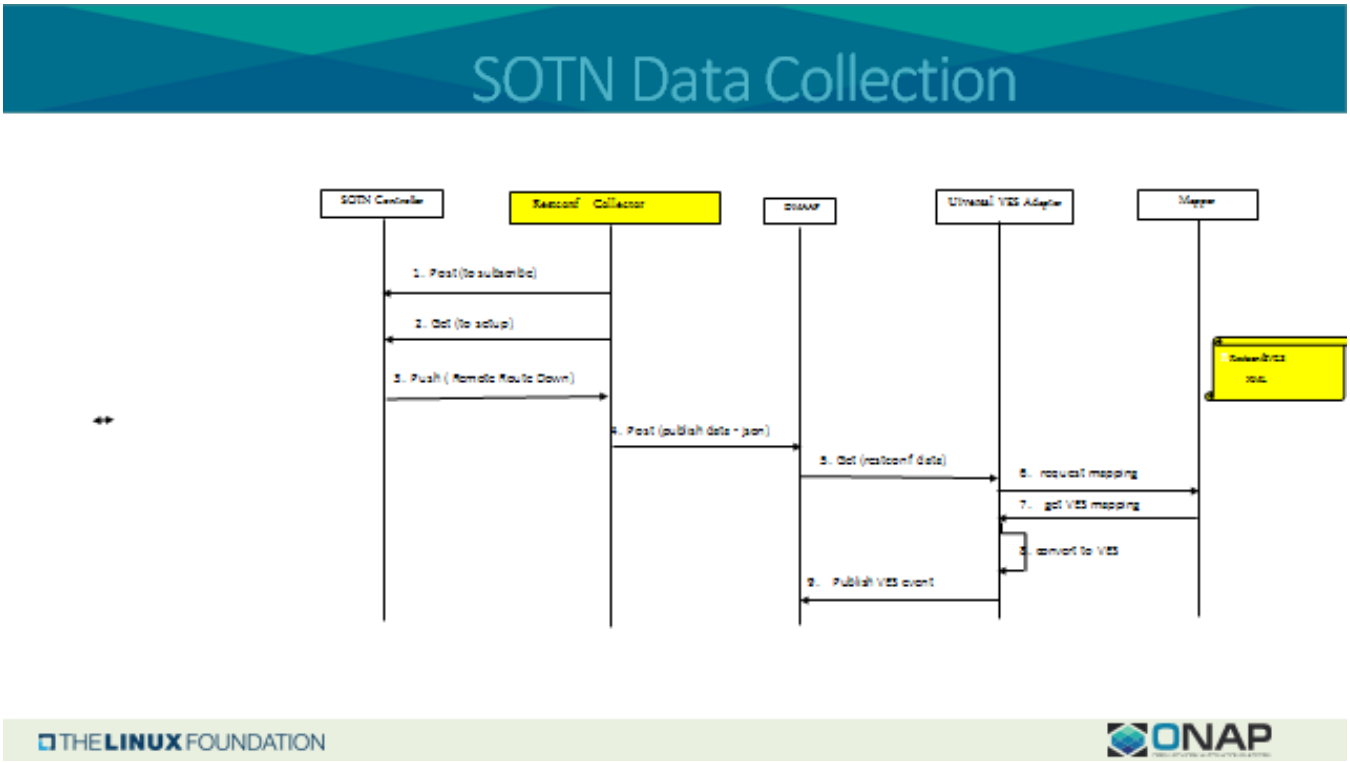
Currently collector's are not deployed on-demand, they are pre-deployed in DCAE and used by all the mS and all control loops. So the collector's, currently, are not deployed via CLAMP and there is no cloudify blueprint available to do so anyway. The collector box in CLAMP GUI is just a virtual box to show the control loop flow, in reality there is no configuration made by CLAMP for the collector (at least at this stage of ONAP, it might change in the future).

This raises the following requirements driven by CCVPN use case towards to design of control loop/CLAMP to accomplish micro-service deployed on demand at CL instantiation.

1. Collector on demand per control loop instance
2. Or, common APIs of collector for updating collector at run time per control loop instantiation
3. One instance of Holmes engine per control loop
4. Configurable policy/actor/recipe mapping at PK

DCAE

DCAE Diagram



The APIs between SOTN Controller and the alarm notification collector following IETF-RESTCONF/YANG-PUSH notification standard:

1. RestConf Collector (RC)subscribes for remote failure alarm to SOTN Controller (SC)
2. RC requests to set up a persistent connection with the 3rd party SC.
3. As the connection is standing, SC pushes service route status data to the collector
4. RC receives alarm data, converts it into JSON format and publishes on DMAAP with topic of ROUTE_ALARM_OUTPUT

5. UVA consumes the alarm message
6. UVA requests the RestConf2VES mapping from the mapper. As of CCVPN use case proposal, the RestConf2VES.xml is manually uploaded to the mapper.
7. Mapper response back the mapping
8. UVA converts json alarm into VES event
9. UVA publishes the VES event on DMAAP for further correlation

APIs to SOTN Controller (Restconf)

Subscribe Notification

Functionality

Collector (Client) and SOTN Controller establishes subscription relationship.

Method

POST

Request-URL

/restconf/operations/ietf-subscribed-notifications:establish-subscription

Request-Body

```
{
  "ietf-subscribed-notifications:input": {
            "encoding": "encode-json"
  }
}
```

Response-Body

```
{
   "ietf-subscribed-notifications:output": {
        "identifier": "1"
   }
}
```

Establish Persistent Connection

Functionality

Collector establishes persistent connection with SOTN Controller and the controller will continuously Push the subscribed notification over.

Method

GET

Request-URL

/restconf/streams/yang-push-json

Request-Body

Response-Body

```
{
  "ietf-notification:notification": {
      "eventTime": {eventTime},
      "ietf-yang-push: push-change-update ": {
          "subscription-id ": {subscription-id },
           " datastore-changes ": {
               "ietf-yang-patch:yang-patch": {
```

```
                "patch-id": {patch-id},
                "edit": [
                    {
                    "edit-id" : {edit-id },
                    "operation" : {operation },
                    "target" : {target },
                    "value": {value}
                   }
                ]
            }
        }
    }
  }
}
```

Restconf Notification

This is the Termination Point (TP) / Route status notification example from SOTN Controller to Restconf data collector, where "eventTime", "tp-id" and "oper-status" will be used in downstream logic.

Service Down Alarm


```
  "ietf-restconf:notification": {
    "eventTime": "2018-07-28T09:15:03.924Z",
    "ietf-yang-push:push-change-update": {
      "subscription-id": 1,
      "datastore-changes": {
        "ietf-yang-patch:yang-patch": {
          "patch-id": "d1d08ce8-b24d-4efb-a0e7-7b835642f2f1",
          "edit": [
            {
              "edit-id": "0",
              "operation": "merge",
              "target": "/network=providerId%2F5555%2FclientId%2F6666%2FtopologyId%2F100/node=example-node/ietf-
network-topology:termination-point=1234",
              "value": {
                "ietf-network-topology:termination-point": [
                  {
                    "supporting-termination-point": [
                      {
                        "network-ref": "providerId/5555/clientId/6666/topologyId/33",
                        "node-ref": "node-ref-example",
                        "tp-ref": "33488898"
                      }
                    ],
                    "ietf-eth-te-topology:svc": {
                      "client-facing": "true",
                      "supported-classification": {
                        "transparent": "true"
                      }
                    },
                    "ietf-te-topology:te": {
                      "admin-status": "up",
                      "interface-switching-capability": [
                        {
                          "encoding": "lsp-encoding-ethernet",
                          "max-lsp-bandwidth": [
                            {
                              "priority": "7",
                              "te-bandwidth": {
                                "ietf-eth-te-topology:eth-bandwidth": "1000000"
                              }
                            }
                          ],
                          "switching-capability": "switching-l2sc"
                        }
                      ],
                      "oper-status": "down",
                      "inter-domain-plug-id": "51000"
                    },
                    "ietf-te-topology:te-tp-id": "1234",
                    "tp-id": "1234"
                  }
                ]
              }
            }
          ]
        }
      }
    }
  }
```


Route Alarm Message

Restconf collector adds one attribute, "notify_oid", to the above JSON data, and publishes it to DMAAP topic RESTCONF_ALARM_TOPIC. This added attribute is used by Universal VES Adapter (Mapper) to identify that this JSON data is from Restconf.

Questions (To be answered):

1. Universal VES Adapter reads all data from a single DMaaP topic, which is specified in / UniversalVesAdapter / src / main / resources / application.properties and defined in / UniversalVesAdapter / src / main / resources / dme2 / consumer.properties. We should publish Restconf data to the general topic, instead of a specific one, RESTCONF_ALARM_TOPIC.
2. If we do publish the data to a Restconf specific topic, Mapper should be tell that the data is from Restconf, and we do not need to add this "notify_oid" attribute.

Restconf Collector Output (JSON)


```
{
  "notify_oid" : "example-mappingfile-id.x.1",

  "ietf-restconf:notification": {
    "eventTime": "2018-07-28T09:15:03.924Z",
    "ietf-yang-push:push-change-update": {
      "subscription-id": 1,
      "datastore-changes": {
        "ietf-yang-patch:yang-patch": {
          "patch-id": "d1d08ce8-b24d-4efb-a0e7-7b835642f2f1",
          "edit": [
            {
              "edit-id": "0",
              "operation": "merge",
              "target": "/network=providerId%2F5555%2FclientId%2F6666%2FtopologyId%2F100/node=example-node/ietf-
network-topology:termination-point=1234",
              "value": {
                "ietf-network-topology:termination-point": [
                  {
                    "supporting-termination-point": [
                      {
                        "network-ref": "providerId/5555/clientId/6666/topologyId/33",
                        "node-ref": "node-ref-example",
                        "tp-ref": "33488898"
                      }
                    ],
                    "ietf-eth-te-topology:svc": {
                      "client-facing": "true",
                      "supported-classification": {
                        "transparent": "true"
                      }
                    },
                    "ietf-te-topology:te": {
                      "admin-status": "up",
                      "interface-switching-capability": [
                        {
                          "encoding": "lsp-encoding-ethernet",
                          "max-lsp-bandwidth": [
                            {
                              "priority": "7",
                              "te-bandwidth": {
                                "ietf-eth-te-topology:eth-bandwidth": "1000000"
                              }
                            }
                          ],
                          "switching-capability": "switching-l2sc"
                        }
                      ],
                      "oper-status": "down",
                      "inter-domain-plug-id": "51000"
                    },
                    "ietf-te-topology:te-tp-id": "1234",
                    "tp-id": "1234"
                  }
                ]
              }
            }
          ]
        }
      }
    }
  }
}
```


Universal VES Adapter (Mapper)

Mapper reads Restconf data from DMaaP topic RESTCONF_ALARM_TOPIC, and uses Smooks to convert it into a standard VES format.

Mapper uses the Restconf2VES.xml to map the Restconf data to VES data. The Restconf2VES.xml could be uploaded to DCAE at the closed loop deployment from DCAE Designer. For now, it will be manually uploaded to the mapper at the closed loop instantiation.

Mapping XML Restconf2VES.xml

Restconf2VES.xml

```
#... for illustration, not a working version

           “varbinds:”:  [{
                 “name”:  “network-ref”,
                 “value”:  “$network-ref”
                  },
                  {
                  “name” : “node-ref",
                  “value”:  “$node-ref”
                  },
                  {
                  “name”: “tp-ref”,
                  “value”:“$tp-ef”,
                  },
                  {
                  “name”:  “te-tp-id”,
                  “value”:  “$te-tp-id”
                  },
                  {
                  “name”:  “tp-id”,
                  “value”:  “$tp-id”
                  },
                  {
                  “name”:  “inter-domain-plug-id”,
                  “value”:  “$inter-domain-plug-id”
                  }
           ]
      }
```

The following is how the resulted VES event data looks like. The event will be published to DMaaP topic unauthenticated.SEC_FAULT_OUTPUT, to be consumed by Holmes.

VES Event

Route Status Alarm (VES/CEDM)


```
{
    "event": {
        "commonEventHeader": {
            "sourceId": "example-target", // 'target' from the restconf notification.
            "startEpochMicrosec": 1413378172000000,
            "eventId": "ab305d54-85b4-a31b-7db2-fb6b977766",
            "sequence": 0,
            "domain": "fault",
            "lastEpochMicrosec": 1413378172000033,
            "eventName": "Fault_Route_Status",
            "sourceName": "example-target", // 'Target' from the restconf notification.
            "priority": "High",
            "version": 3.0,
            "reportingEntityName": "Domain_Contorller"
        },
        "faultFields": {
            "eventSeverity": "CRITICAL",
            "alarmCondition": "Route_Status",
            "faultFieldsVersion": 2.0,
            "specificProblem": "Fault_SOTN_Service_Status",
            "alarmAdditionalInformation": [
                {
                "name": "networkId",
                "value": "providerId%2F5555%2FclientId%2F6666%2FtopologyId%2F100"
                },
                {
                "name": "node",
                "value": "example-node"
                },
                {
                "name": "tp-id",
                "value": "1234"
                },
                {
                "name": "oper-status",
                "value": "down"
                }
            ]
            "eventSourceType": "other",
            "vfStatus": "Active"
        }
    }
}
```


DCAE Blueprints

The initial/default configuration of restconf collector and Holmes are specified in the following micro-service blueprints.

Blueprint for Restconf Collector

Restconf.yaml.template

```
version: '2.1'
services:
   restconfcollector:
      image: "restconfcollector"
      container_name: "restconf_collector"
      restart: "always"
      hostname: "restconf-collector"
   labels:
       - "SERVICE_NAME=restconf_collector"
```

Blueprint for Holmes Rule Manager

Holmes-rules.yaml.template

HOLMES (WIP)

CCVPN Close Loop requires Holmes to correlate route down alarms from SOTN Controllers from different sites. (Refer to this page on Holmes installation,)

In Phase 1, for the minimum, two of the above defined Route_Down_Alarm will be correlated within time window of 15 - 30, for instance, milliseconds. (Refer to this page as examples.)

Rule Creation

Holmes Rule


```
{
    "content":"package org.onap.holmes.ccvpn;\n\ndialect \"java\"\n\nimport org.onap.holmes.common.api.stat.
VesAlarm;\nimport org.onap.holmes.common.api.stat.AlarmAdditionalField;\nimport org.onap.holmes.common.aai.
AaiQuery4Ccvpn;\nimport org.onap.holmes.common.exception.CorrelationException;\nimport org.onap.holmes.common.
dmaap.entity.PolicyMsg;\nimport org.onap.holmes.common.dmaap.DmaapService;\nimport org.onap.holmes.common.utils.
DroolsLog;\nimport org.onap.holmes.common.dropwizard.ioc.utils.ServiceLocatorHolder;\n\nimport com.alibaba.
fastjson.JSONArray;\nimport com.alibaba.fastjson.JSONObject;\n\nimport java.util.List;\nimport java.util.Map;
\nimport java.util.ArrayList;\nimport java.util.HashMap;\nimport java.util.UUID;\n\nfunction String
getAdditionalField(VesAlarm a, String field) {\n        List<AlarmAdditionalField> fields = a.
getAlarmAdditionalInformation();\n        for (AlarmAdditionalField f : fields) {\n                if (f.
getName().equals(field)) {\n                        return f.getValue();\n                }\n        }\n
return null;\n}\n\nfunction String getLogicLink(VesAlarm alarm) {\n        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.
newInstance();\n        return aai.getLogicLink(\n                getAdditionalField(alarm, \"networkId\"),
\n                getAdditionalField(alarm, \"node\"),\n                getAdditionalField(alarm, \"tp-id\"),
\n                null\n        );\n}\n\nfunction boolean isCorrelated(VesAlarm a, VesAlarm b) {\n
String logicLinkA = getLogicLink(a);\n        if (logicLinkA == null)        {\n                return false;
\n        }\n\n        String logicLinkB = getLogicLink(b);\n        if (logicLinkB == null) {\n
return false;\n        }\n\n        return logicLinkA.equals(logicLinkB);\n}\n\nfunction void
updateAaiLinkStatus(String linkName, String status) {\n        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.
newInstance();\n        Map<String, Object> body = new HashMap<String, Object>(){\n
{\n                        put(\"operational-status\", status);\n                }\n        };\n        aai.
updateLogicLinkStatus(linkName, body);\n}\n\nfunction void updateAaiTpStatus(String networkId, String pnfName,
String ifName, String status) {\n        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.newInstance();\n
Map<String, Object> body = new HashMap<String, Object>(){\n                {\n                        put(\"
operational-status\", status);\n                }\n        };\n        aai.updateTerminalPointStatus(networkId,
pnfName, ifName, body);\n}\n\nfunction Map<String, Object> getAdditionalResourceInfo(String networkId, String
pnfName, String ifName, String status) {\n        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.newInstance();\n
JSONArray instances = aai.getServiceInstances(networkId, pnfName, ifName, status);\n\n        Map<String,
Object> ret = new HashMap<String, Object>();\n\n        StringBuilder sbn = new StringBuilder();\n
StringBuilder sbi = new StringBuilder();\n        for(int i = 0; i < instances.size(); ++i) {\n
JSONObject o = instances.getJSONObject(i);\n                String name = o.getString(\"service-instance-
```

```
name\");\n                String id = o.getString(\"service-instance-id\");\n                ret.put(id + \".
input-parameters\", o.getString(\"input-parameters\"));\n                sbn.append(name).append(\",\");
\n                sbi.append(id).append(\",\");\n        }\n        ret.put(\"service-instance.service-instance-
name\", sbn.substring(0, sbn.length() -1).toString());\n        ret.put(\"service-instance.service-instance-
id\", sbi.substring(0, sbi.length() -1).toString());\n        ret.put(\"vserver.vserver-name\", \"TBD\");
\n        ret.put(\"globalSubscriberId\", instances.getJSONObject(0).getString(\"globalSubscriberId\"));
\n        ret.put(\"serviceType\", instances.getJSONObject(0).getString(\"serviceType\"));\n\n        return
ret;\n}\n\nfunction PolicyMsg createPolicyMsg(VesAlarm alarm) {\n        PolicyMsg m = new PolicyMsg();
\n        m.setPolicyVersion(\"1.0.0.5\");\n        m.setPolicyName(\"CCVPN\");\n        m.setPolicyScope(\"
service=SOTNService,type=SampleType,closedLoopControlName=CL-CCVPN-d925ed73-8231-4d02-9545-db4e101f88f8\");
\n        m.setClosedLoopControlName(DmaapService.loopControlNames.get(\"org.onap.holmes.ccvpn\"));\n        m.
setRequestID(UUID.randomUUID().toString());\n        m.setClosedLoopAlarmStart(alarm.getStartEpochMicrosec());
\n        m.setClosedLoopAlarmEnd(alarm.getLastEpochMicrosec());\n        m.setTarget(\"vserver.vserver-name\");
\n        m.setAai(getAdditionalResourceInfo(\n                getAdditionalField(alarm, \"networkId\"),
\n                getAdditionalField(alarm, \"node\"),\n                getAdditionalField(alarm, \"tp-id\"),
\n                getAdditionalField(alarm, \"oper-status\")\n        ));\n\n        DmaapService.
alarmUniqueRequestID.put(alarm.getEventId(), m.getRequestID());\n\n        return m;\n}\n\nrule \"Update AAI
Information\"\n        no-loop true\n        salience 300\n        when\n                $a: VesAlarm(eventName.
indexOf(\"Fault_Route_Status\") != -1)\n        then\n                updateAaiTpStatus
(\n                        getAdditionalField($a, \"networkId\"),\n                        getAdditionalField
($a, \"node\"),\n                        getAdditionalField($a, \"tp-id\"),\n
getAdditionalField($a, \"oper-status\")\n                );\nend\n\nrule \"Set Up Correlation\"\n        no-
loop true\n        salience 200\n        when\n                $a: VesAlarm($id: eventId,
\n                                        $start: startEpochMicrosec,
\n                                        eventName.indexOf(\"Fault_Route_Status\") != -1)
\n                $b: VesAlarm(eventId != $id, \n                                        eventName.
indexOf(\"Fault_Route_Status\") != -1, \n                                        Math.abs
(startEpochMicrosec - $start) < 60000)\n        then\n                String status = \"down\";
\n                if (status.equalsIgnoreCase(getAdditionalField($a, \"oper-status\"))
\n                        && status.equalsIgnoreCase(getAdditionalField($b, \"oper-status\")))
{\n                if (isCorrelated($a, $b)){\n                        // If any of the alarms have
been marked as root, a policy message has ever been created and sent. Do NOT send it again.
\n                        if ($a.getRootFlag() != 1 && $b.getRootFlag() != 1)
{\n                                PolicyMsg msg = createPolicyMsg($a);\n
DmaapService dmaapService = ServiceLocatorHolder.getLocator().getService(DmaapService.class);
\n                        dmaapService.publishPolicyMsg(msg, \"unauthenticated.DCAE_CL_OUTPUT\");
\n                        updateAaiLinkStatus(getLogicLink($a), status);\n                }
\n                $a.setRootFlag(1);\n                $b.setRootFlag(1);
\n                update($a);\n                update($b);\n                }\n        }
\nend\n\nrule \"Clear Alarms\"\n        no-loop true\n        salience 100\n        when\n        $a:
VesAlarm(eventName.indexOf(\"Fault_Route_Status\") != -1)\n        then\n                if (\"up\".
equalsIgnoreCase(getAdditionalField($a, \"oper-status\"))) {\n                if (DmaapService.
alarmUniqueRequestID.containsKey($a.getEventId())) {\n                DmaapService.
alarmUniqueRequestID.remove($a.getEventId());\n                }\n        \n
//TODO: send alarm clearing message to Policy - for now it's not needed.\n        //...\n
\n                retract($a);\n        }\nend\n",
    "description":"This rule is designed for the correlation analysis for the CCVPN use case.",
    "enabled":1,
    "loopControlName":"ControlLoop-CCVPN-2179b738-fd36-4843-a71a-a8c24c70c55b",
    "ruleName":"CCVPN"
}
```


To illustrate, the content field in the above rule is presented as readable format in the following. We need to convert it to valid json string when uploading to Holmes.

Holmes Rule Content

```
package org.onap.holmes.ccvpn;

dialect "java"

import org.onap.holmes.common.api.stat.VesAlarm;
import org.onap.holmes.common.api.stat.AlarmAdditionalField;
import org.onap.holmes.common.aai.AaiQuery4Ccvpn;
import org.onap.holmes.common.exception.CorrelationException;
import org.onap.holmes.common.dmaap.entity.PolicyMsg;
```

```
import org.onap.holmes.common.dmaap.DmaapService;
import org.onap.holmes.common.utils.DroolsLog;
import org.onap.holmes.common.dropwizard.ioc.utils.ServiceLocatorHolder;

import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;

import java.util.List;
import java.util.Map;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.UUID;

function String getAdditionalField(VesAlarm a, String field) {
        List<AlarmAdditionalField> fields = a.getAlarmAdditionalInformation();
        for (AlarmAdditionalField f : fields) {
                if (f.getName().equals(field)) {
                        return f.getValue();
                }
        }
        return null;
}

function String getLogicLink(VesAlarm alarm) {
        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.newInstance();
        return aai.getLogicLink(
                getAdditionalField(alarm, "networkId"),
                getAdditionalField(alarm, "node"),
                getAdditionalField(alarm, "tp-id"),
                null
        );
}

function boolean isCorrelated(VesAlarm a, VesAlarm b) {
        String logicLinkA = getLogicLink(a);
        if (logicLinkA == null)        {
                return false;
        }

        String logicLinkB = getLogicLink(b);
        if (logicLinkB == null) {
                return false;
        }

        return logicLinkA.equals(logicLinkB);
}

function void updateAaiLinkStatus(String linkName, String status) {
        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.newInstance();
        Map<String, Object> body = new HashMap<String, Object>(){
                {
                        put("operational-status", status);
                }
        };
        aai.updateLogicLinkStatus(linkName, body);
}

function void updateAaiTpStatus(String networkId, String pnfName, String ifName, String status) {
        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.newInstance();
        Map<String, Object> body = new HashMap<String, Object>(){
                {
                        put("operational-status", status);
                }
        };
        aai.updateTerminalPointStatus(networkId, pnfName, ifName, body);
}

function Map<String, Object> getAdditionalResourceInfo(String networkId, String pnfName, String ifName, String
status) {
        AaiQuery4Ccvpn aai = AaiQuery4Ccvpn.newInstance();
        JSONArray instances = aai.getServiceInstances(networkId, pnfName, ifName, status);
```

```
        Map<String, Object> ret = new HashMap<String, Object>();

        StringBuilder sbn = new StringBuilder();
        StringBuilder sbi = new StringBuilder();
        for(int i = 0; i < instances.size(); ++i) {
                JSONObject o = instances.getJSONObject(i);
                String name = o.getString("service-instance-name");
                String id = o.getString("service-instance-id");
                ret.put(id + ".input-parameters", o.getString("input-parameters"));
                sbn.append(name).append(",");
                sbi.append(id).append(",");
        }
        ret.put("service-instance.service-instance-name", sbn.substring(0, sbn.length() -1).toString());
        ret.put("service-instance.service-instance-id", sbi.substring(0, sbi.length() -1).toString());
        ret.put("vserver.vserver-name", "TBD");
        ret.put("globalSubscriberId", instances.getJSONObject(0).getString("globalSubscriberId"));
        ret.put("serviceType", instances.getJSONObject(0).getString("serviceType"));

        return ret;
}

function PolicyMsg createPolicyMsg(VesAlarm alarm) {
        PolicyMsg m = new PolicyMsg();
        m.setPolicyVersion("1.0.0.5");
        m.setPolicyName("CCVPN");
        m.setPolicyScope("service=SOTNService,type=SampleType,closedLoopControlName=CL-CCVPN-d925ed73-8231-4d02-
9545-db4e101f88f8");
        m.setClosedLoopControlName(DmaapService.loopControlNames.get("org.onap.holmes.ccvpn"));
        m.setRequestID(UUID.randomUUID().toString());
        m.setClosedLoopAlarmStart(alarm.getStartEpochMicrosec());
        m.setClosedLoopAlarmEnd(alarm.getLastEpochMicrosec());
        m.setTarget("vserver.vserver-name");
        m.setAai(getAdditionalResourceInfo(
                getAdditionalField(alarm, "networkId"),
                getAdditionalField(alarm, "node"),
                getAdditionalField(alarm, "tp-id"),
                getAdditionalField(alarm, "oper-status")
        ));

        DmaapService.alarmUniqueRequestID.put(alarm.getEventId(), m.getRequestID());

        return m;
}

rule "Update AAI Information"
        no-loop true
        salience 300
        when
                $a: VesAlarm(eventName.indexOf("Fault_Route_Status") != -1)
        then
                updateAaiTpStatus (
                        getAdditionalField($a, "networkId"),
                        getAdditionalField($a, "node"),
                        getAdditionalField($a, "tp-id"),
                        getAdditionalField($a, "oper-status")
                );
end

rule "Set Up Correlation"
        no-loop true
        salience 200
        when
                $a: VesAlarm($id: eventId,
                                                $start: startEpochMicrosec,
                                                eventName.indexOf("Fault_Route_Status") != -1)
                $b: VesAlarm(eventId != $id,
                                                eventName.indexOf("Fault_Route_Status") != -1,
                                                Math.abs(startEpochMicrosec - $start) < 60000)
        then
                String status = "down";
```

```
                if (status.equalsIgnoreCase(getAdditionalField($a, "oper-status"))
                                && status.equalsIgnoreCase(getAdditionalField($b, "oper-status"))) {
                    if (isCorrelated($a, $b)){
                            // If any of the alarms have been marked as root, a policy message has ever been
created and sent. Do NOT send it again.
                            if ($a.getRootFlag() != 1 && $b.getRootFlag() != 1) {
                                    PolicyMsg msg = createPolicyMsg($a);
                                DmaapService dmaapService = ServiceLocatorHolder.getLocator().getService
(DmaapService.class);
                                dmaapService.publishPolicyMsg(msg, "unauthenticated.DCAE_CL_OUTPUT");
                                updateAaiLinkStatus(getLogicLink($a), status);
                            }
                        $a.setRootFlag(1);
                        $b.setRootFlag(1);
                        update($a);
                        update($b);
                    }
            }
end

rule "Clear Alarms"
        no-loop true
        salience 100
        when
                $a: VesAlarm(eventName.indexOf("Fault_Route_Status") != -1)
        then
                if ("up".equalsIgnoreCase(getAdditionalField($a, "oper-status"))) {
                        if (DmaapService.alarmUniqueRequestID.containsKey($a.getEventId())) {
                                DmaapService.alarmUniqueRequestID.remove($a.getEventId());
                        }

                        //TODO: send alarm clearing message to Policy - for now it's not needed.
                //...

                        retract($a);
                }
end
```

Rule Execution

After the correlation is done successfully, there should be a corresponding control loop event defined in the following section published on the unauthenticated.DCAE_CL_OUTPUT topic of DMaaP.

POLICY

Policy Creation

Operational Policy - Yaml

```
controlLoop:
  version: 2.0.0
  controlLoopName: ControlLoop-CCVPN-2179b738-fd36-4843-a71a-a8c24c70c66b
  trigger_policy: unique-policy-id-16-Reroute
  timeout: 3600
  abatement: false

policies:
  - id: unique-policy-id-16-Reroute
    name: Connectivity Reroute
    description:
    actor: SDNC
    recipe: Reroute
    target:
      type: VM
    retry: 3
    timeout: 1200
    success: final_success
    failure: final_failure
    failure_timeout: final_failure_timeout
    failure_retries: final_failure_retries
    failure_exception: final_failure_exception
    failure_guard: final_failure_guard
```

For now, Policy bases on the parameters looked up and encoded in by Holmes to the 'AAI' portion of the following event to invoke SDNC API to re-route the network connectivity.

DCAE Control Loop Event (Holmes)

```
{
    "closedLoopEventClient": "DCAE.HolmesInstance",
    "policyVersion": "1.0.0.5",
    "policyName": "CCVPN",
    "policyScope": "service=SOTNService,type=SampleType,closedLoopControlName=CL-CCVPN-d925ed73-8231-4d02-9545-
db4e101f88f8",
    "target_type": "VM",
    "AAI": {
        "vserver.vserver-name" : "TBD",
                "globalSubscriberId" : "e151059a-d924-4629-845f-264db19e50b4",
                "serviceType" : "SOTN",
        "service-information.service-instance-id" : "service-instance-id-example-1",
        "network-information. network-id " : "id"
    },
    "closedLoopAlarmStart": 1484677482204798,
    "closedLoopEventStatus": "ONSET",
    "closedLoopControlName": "ControlLoop-CCVPN-2179b738-fd36-4843-a71a-a8c24c70c66b",
    "version": "1.0.2",
    "target": "vserver.vserver-name",
    "requestID": "97964e10-686e-4790-8c45-bdfa61df770f",
    "from": "DCAE"
}
```

Policy Engine subscribes the unauthenticated.DCAE_CL_OUTPUT on DMAAP.

AAI Enrichment APIs

DCAE VES event A&AI enrichment for previous use cases are defined here as reference. More CCVPN related AAI API discussion can be found on this page,

Note: pnfName = Nodeid and p-interface-name = tp-id

- The query logic from tp-id through service instance is that : p-interface vpn-vpnbinding connectivity service instance. Then from service-instance to 'customer-request' for service instance recreation for phase 1.
- For event from different tp-id to be correlated: p-interface logical link, then logic link is the same from ONAP domain.
- When link down event is detected, Closed Loop needs to update logical-link/p-interface's operational-status to “DOWN” for the recreated service instance picking up new links.

The following APIs are used to support the above looking up.

Status Update

Update TP Status

```
URL: https://<AAI host>:<AAI port>/aai/v14/network/network-resources/network-resource/{networkId}/pnfs/pnf/
{pnfName}/p-interfaces/p-interface/{ifName}
Method: PATCH
Request Body:
{
    "operational-status": "some status"
}
```

Update Logic-link Status

```
URL: https://<AAI host>:<AAI port>/aai/v14/network/logical-links/logical-link/{linkName}
Method: PATCH
Request Body:
{
    "operational-status": "some status"
}
```

Service Instance ID Look Up

Get vpn-binding from TP

```
URL:
https://<AAI host>:<AAI port>/aai/v14/network/network-resources/network-resource/{networkId}/pnfs/pnf/{pnfName}
/p-interfaces?interface-name={ifName}&operational-status={status}
Method: GET
Request Body:
{
}

Response Body:
{
    "results": [
      {
        "p-interface": {
          "interface-name": "{ifName}",
          "network-ref": "some ref",
          "transparent": "some value",
          "operational-status ": "{status}",
          "speed-value": "some speed",
          "relationship-list": {
              "relationship" : [
                  {
                      "related-to": "vpn-binding",
                      "related-link": "url of vpn-binding",
                      "relationship-data": [
                         "relationship-key": "vpn-binding.vpn-id",
                         "relationship-value": "some id"
                      ]
                  }
              ]
          }
        }
      }
    ]
}
```

Get connectivity from vpn-binding

```
URL: https://<AAI host>:<AAI port>/aai/v14/network/vpn-bindings?vpn-id={vpnId}
Method: GET
Request Body:
{
}

Response Body:
{
    "results": [
      {
        "vpn-binding": {
          "vpn-id": "{vpnId}",
          "vpn-name": "some name",
          "access-provider-id": "provider id",
          "access-client-id": "client id",
          "access-topology-id": "topology id",
          "src-access-node-id": "src node id",
          "src-access-ltp-id": "src ltp id",
          "dst-access-node-id": "dst node id",
          "dst-access-ltp-id": "dst ltp id,
          "operational-status": "some status",
          "relationship-list": {
              "relationship" : [
                  {
                      "related-to": "connectivity",
                      "related-link": "url of connectivity",
                      "relationship-data": [
                         "relationship-key": "connectivity. connectivity-id",
                                                 "relationship-value": "some id"
                      ]
                  }
              ]
            }
        }
      }
    ]
}
```

Query service instances by connectivity

```
URL: https://<AAI host>:<AAI port>/aai/v14/network/connectivities/connectivity/{connectivityId}
Method: Get
Request Body:
{
}
Response Body:
{
    "results": [
      {
        "connectivity": {
          "connectivity-id": "{connectivityId}",
          "bandwidth-profile-name": "some profile",
          "vpn-type": "some type",
          "cir": "cir value",
          "eir": "eir value",
          "cbs": "cbs value",
          "ebs": "ebs value",
          "color-aware": "color value",
          "coupling-flag": "flag value",
          "etht-svc-name": "some name",
          "access-provider-id": "provider id",
          "access-client-id": "client id",
          "access-topology-id": "topology id",
          "access-node-id": "node id",
          "access-ltp-id": "ltp id",
          "connectivity-selflink": "some URL",
          "cvlan ": "some tag",
          "operational-status": "some status",
          "relationship-list": {
              "relationship" : [
                  {
                      "related-to": "service-instance",
                      "related-link": "url of service-instance",
                      "relationship-data": [
                                                          "relationship-key": "service-instance.service-instance-
id",
                                                          "relationship-value": "some id"
                      ]
                  }
              ]
          }
      }
    ]
```

Get all 3 services instances from CCVPN

```
From service-instance URL:

URL: https://<AAI host>:<AAI port>/aai/v14/business/customers/customer/{global-customer-id}/service-
subscriptions/service-subscription/{service-type}/service-instances?service-instance-id={servId}

Derive the service-subscription URL:
URL: https://<AAI host>:<AAI port>/aai/v14/business/customers/customer/{global-customer-id}/service-
subscriptions/service-subscription/{service-type}


Method: GET
Request Body:
{
}
Response Body:
{
    "results": [
      {
        "service-subscription": {
          "service-type": "{service-type}",
          "temp-ub-sub-account-id": "some sub account",
          "service-instances": {
              "service-instance": [
                {
                  "service-instance-id": "some id 1",
                  "service-instance-name": "some name 1",
                  "environment-context": "some context 1",
                  "workload-context": "some workload 1",
                  "relationship-list": {
                      "relationship" : [
                      ]
                  }
                },
                {
                  "service-instance-id": "some id 2",
                  "service-instance-name": "some name 2",
                  "environment-context": "some context 2",
                  "workload-context": "some workload 2",
                  "relationship-list": {
                      "relationship" : [
                      ]
                  }
                },
                {
                  "service-instance-id": "some id 3",
                  "service-instance-name": "some name 3",
                  "environment-context": "some context 3",
                  "workload-context": "some workload 3",
                  "relationship-list": {
                      "relationship" : [
                      ]
                  }
                }
              ]
          },
          "relationship-list": {
              "relationship" : [
              ]
          }
        }
      }
    ]
}


For each
item in results: -   Get item.service-subscription.service-instances
        -   For each data in service-instances:
        -   Get service-instance object
```

Alarm Correlation

Get Logic-link from TP

```
URL: https://<AAI host>:<AAI port>/aai/v14/network/pnfs/pnf/{pnfName}/p-interfaces?interface-name={ifName}
&operational-status={status}
Method: GET
Resquest Body:
{
}


Response Body:
{
        "results" : [
                {
        "p-interface" : {
             "interface-name":{ifName}",
                         "network-ref": "some ref",
             "transparent": "some blue",
             "operational-status":"{status}",
             "speed-value" : "some speed",
             "relationship-list":
                "relationship" : [
                    {
                      "related-to" : "logic-link",
                      "related-link" :'url of logical-link",
                      "relationship-data":[
                          "relationship-key" : "logical-link.link.name",
                          "relationship-value"; "some name"
                      ]
                    }
                ]
            }
    ]
}
```

Look up for 'input-parameters' by 'service-instance-id'

Query service instances for CCVPN

```
URL: https://<AAI host>:<AAI port>/aai/v14/business/customers/customer/{global-custoner-id}/service-
subscriptions/service-subscription/{service-type}/service-instances?service-instance-id={serviceId}
Method: GET
Request Body:
{
}
Response Body:
{
        "service-instance-id": "{service-instance-id}",
    "service-instance-name" : "instance name",
        "service-type": "some type",
        "service-role": "some role",
        "model-invariant-id": "model id",
        "model-version-id": "model version",
        "input-parameters:"request parameters",  // ... This is the service instance recreation input looked up
by CL.
        "resource-version": "some version"
}


Example of response body:
{
        "service-instance-id": "176d9eba-1662-4289-8396-0097b50fd485",
        "service-type": "E2E Service",
        "service-role": "E2E Service",
        "model-invariant-id": "c22a9483-d2b6-49cc-b1f7-ef34c93572a1",
        "model-version-id": "71d0e396-e246-4c23-aa57-6da2043d6209",
        "input-parameters:".....;",             // ... This is the service instance recreation input looked up
by CL.
        "resource-version": "1528975017336"
        "relationship-list": {
             "relationship": [
                  {
                  "related-to": "pnf",
                  "related-link": "/aai/v11/network/pnfs/pnf/MME-0001",
                  "relationship-data": [
                                  {
                        "relationship-key": "pnf.pnf-id",
                        "relationship-value": "176d9eba-1662-4289-8396-0097b50fd466"
                                  }
                             ],
                  "related-to-property": [
                                      {
                                      "property-key": "pnf.pnf-name",
                                      "property-value": "MME-0001"
                                  }
                             ]
                        }
                ]
        }
}
```

Others

Query for Logical-links

```
URL: https://<AAI host>:<AAI port>/aai/v14/network/logical-links?link-name={linkName}&operational-status=
{status}
Method: GET
{
}

Response Body:
{
"results": [
    "logic-links" : {
      "link-name" : "{linkName}",
      "operational-status": "{status}",
      "model-invariant-id": "some invariant",
      "model-version-id"  : "some version",
      "link-id":"some id",
      "relationship-list"  : [
            {
            "relationship" : [
                {
                     "related-to": "p-interface",
                     "related-link": "url of p-interface 1",
                     "relationship-data":[
                          "relationship-key" : "p-interface.interface-name",
                          "relationship-value" : "some name 1"
                      ]
                }
             ],
           },
           {
            "relationship" : [
                {
                     "related-to": "p-interface",
                     "related-link": "url of p-interface 2",
                     "relationship-data":[
                          "relationship-key" : "p-interface.interface-name",
                          "relationship-value" : "some name 2"
                      ]
                }
               ],
            }
      ]
 }
```