HPE UCA Automation

Integrator Guide for Linux (RHEL 6.4)

Version: 2.1
Edition: 1.0
July 2016

Hewlett Packard
Enterprise

Notices

Contents

List of figures

List of tables

Preface

About this guide

This guide provides an overview of the UCA Automation product and describes how to create value packs for specific domain specializations and integrate them with the UCA Automation product.

Product Name: UCA Automation

Product Version: 2.1

Read this document before installing or using this software.

Audience

This guide is intended for system integrators, solution developers, and software development engineers.

Software versions

The term UNIX is used as a generic reference to the operating system, unless otherwise specified.

The software versions referred to in this document are as follows:

Table 1: Software versions

Product Version	Supported Operating systems
UCA Automation 2.1	Linux Red Hat Enterprise Linux Server release RHEL 6.4

Typographical conventions

`Fixed width text`	It is used for filenames and their contents, computer inputs or outputs, program codes, and so on.
Italic text	It is used for labels, parameters, emphasized text, and replaceable text, citations and references
Bold text	It is used to indicate navigation options in the interfaces; for example, the text appearing in buttons and menu items. User interface controls, window titles, generic emphasis
<angle brackets>	Indicates generic variable names that must be substituted by real values or strings.

Reference Documents

- UCA Automation Installation Guide
- UCA Automation Administrator and User Interface Guide

- UCA EBC Problem Detection Installation Administration and Dev Guide
- UCA for Event Based Correlation Value Pack Development Guide
- HPE Service Activator Overview Guide
- HPE Service Activator Putting Service Activator To Work Guide
- HPE Service Activator Plug-ins Guide

Support

Please visit our HPE Software Support Online Web site at softwaresupport.hpe.com for contact information, and details about HPE Software products, services, and support.

The software support area of the software web site includes the following:

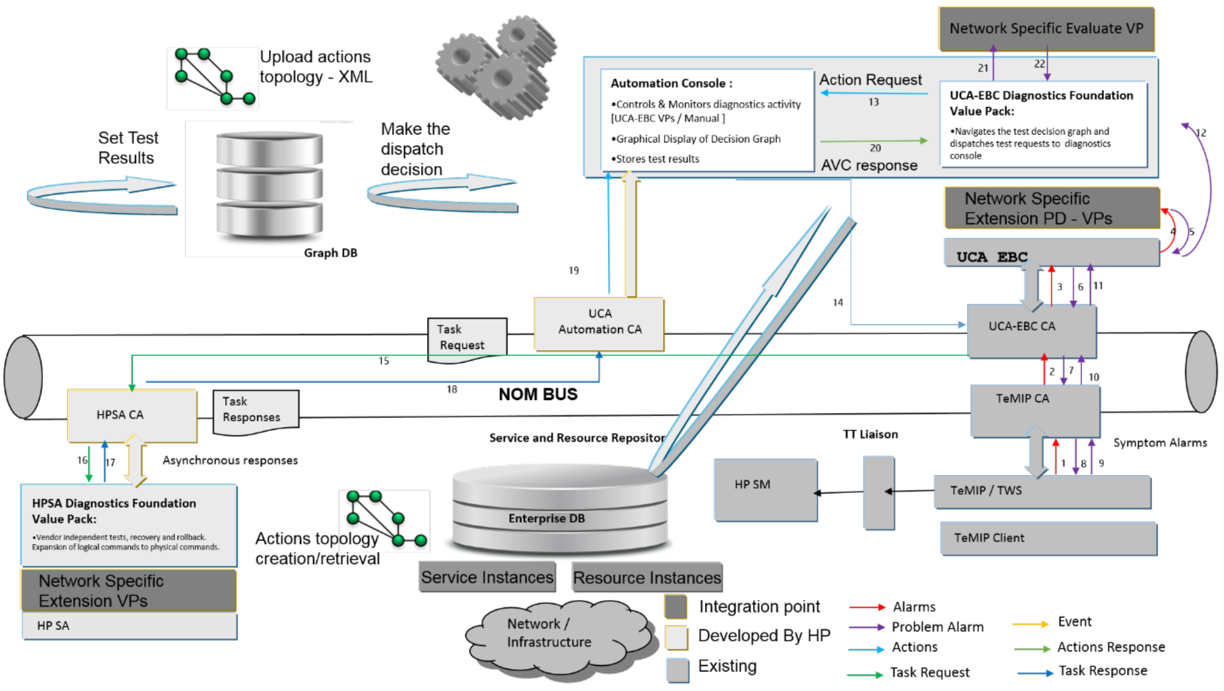
- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

Chapter 1 Introduction

The UCA Automation system is a platform for building value added resolution automations based on a combination of business rules and workflow rules. UCA Automation isolates network related issues and automates the corresponding resolutions.

The UCA Automation software is a combination of business rules engine and workflows engine. The system combines HPE Unified Correlation Analyzer for Event Based Correlation (UCA EBC) system, which provides the business rules capability with HPE Service Activator (HPE SA), which provides the activation capability, through the enterprise service bus called NGOSS Open Mediation (NOM) and Unified Mediation Bus (UMB).

The following diagram shows the architecture of the UCA Automation system with NOM.



Upload actions topology - XML
Set Test Results
Make the dispatch decision
Graph DB
Network Specific Evaluate VP
21
22
Automation Console :
•Controls & Monitors diagnostics activity [UCA-EBC VPs / Manual]
•Graphical Display of Decision Graph
•Stores test results
Action Request
13
20
AVC response
UCA-EBC Diagnostics Foundation Value Pack:
•Navigates the test decision graph and dispatches test requests to diagnostics console
12
Network Specific Extension PD - VPs
4
5
UCA EBC
3
6
11
19
14
UCA Automation CA
Task Request
UCA-EBC CA
15
18
NOM BUS
2
7
10
TeMIP CA
HPSA CA
Task Responses
Service and Resource Repositor
TT Liaison
Symptom Alarms
16
17
Asynchronous responses
1
8
9
HP SM
TeMIP / TWS
Enterprise DB
HPSA Diagnostics Foundation Value Pack:
•Vendor independent tests, recovery and rollback. Expansion of logical commands to physical commands.
Actions topology creation/retrieval
TeMIP Client
Network Specific Extension VPs
Service Instances
Resource Instances
HP SA
Network / Infrastructure
Integration point
Developed By HP
Existing
Alarms
Problem Alarm
Actions
Task Request
Event
Actions Response
Task Response

Figure 1: UCA Automation with NOM

The following diagram shows the architecture of the UCA Automation system with UMB.

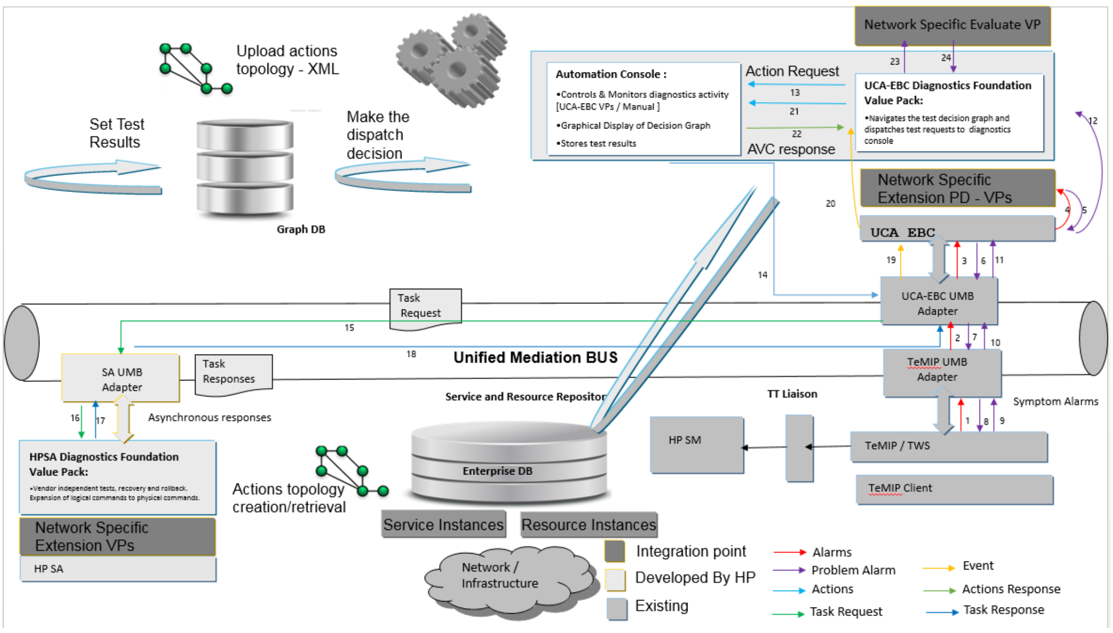

Upload actions topology - XML
Set Test Results
Make the dispatch decision
Graph DB
Network Specific Evaluate VP
Automation Console :
•Controls & Monitors diagnostics activity [UCA-EBC VPs / Manual]
•Graphical Display of Decision Graph
•Stores test results
Action Request
13
21
22
AVC response
23
24
UCA-EBC Diagnostics Foundation Value Pack:
•Navigates the test decision graph and dispatches test requests to diagnostics console
12
Network Specific Extension PD - VPs
20
4
5
UCA EBC
19
3
6
11
14
Task Request
UCA-EBC UMB Adapter
15
2
7
10
18
Unified Mediation BUS
SA UMB Adapter
Task Responses
TeMIP UMB Adapter
Symptom Alarms
Service and Resource Repository
TT Liaison
16
17
Asynchronous responses
HP SM
1
8
9
TeMIP / TWS
HPSA Diagnostics Foundation Value Pack:
•Vendor independent tests, recovery and rollback. Expansion of logical commands to physical commands.
Enterprise DB
TeMIP Client
Actions topology creation/retrieval
Network Specific Extension VPs
Service Instances
Resource Instances
HP SA
Network / Infrastructure
Integration point
Developed By HP
Existing
Alarms
Problem Alarm
Actions
Task Request
Event
Actions Response
Task Response

Figure 2: UCA Automation with UMB

The grey blocks are the Network Specific Extension Value Packs, optionally Network Specific Evaluate Value Pack, and HPE SA Network Specific Extension Value Packs should be implemented to create and integrate custom automations.

1.1 Design theory

Two key functions are performed by UCA Automation; the problem isolation and problem resolution.

Problem isolation is the responsibility of UCA EBC Problem Diagnosis value pack, which can eliminate event storms, false positives, false negatives, and deduce a single meaningful problem alarm.

This information is then passed to the decide-and-act engine, which identifies the action to be taken for a specific problem. After the action, the evolved knowledge is sent back to the decide-and-act engine for further resolution based on the decision tree or evaluate value-pack optionally, to perform predictive and proactive automation. In addition, diagnostic information is gathered automatically to reduce the MTTR (mean time to resolve).

The UCA Automation system works in the way depicted by the following diagram. It starts with the original problem, performs tests after tests as per the decision tree design, and then either resolves the problem or enriches the problem alarm with complete diagnosis, or can even create a trouble ticket automatically.

In case of manual resolution, the operator is presented with a set of problems, the associated services, and a list of the types of devices which can support such services. Once the above triplet is chosen, the corresponding resolutions are displayed, which can be invoked manually.

In UCA Automation System, the process of problem resolution happens in the way depicted by the following diagram. The administrator or integrator of the system has the option to easily configure the decision tree without the need for any kind of programming. The decide and act subsystems work based on this configuration. In case the administrator needs to make advanced decisions based on the results of the previous tests, the platform allows him to write his own rules in the evaluate block.

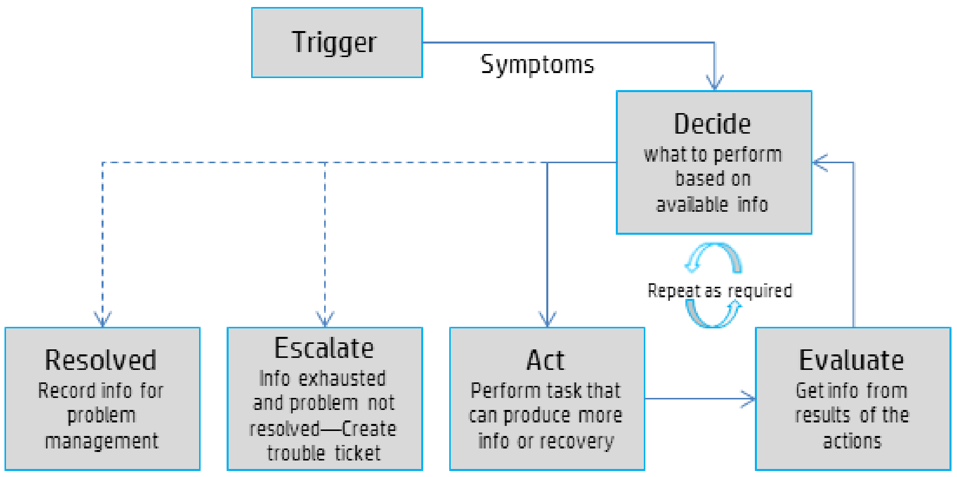

Trigger
Symptoms
Decide
what to perform based on available info
Repeat as required
Resolved
Record info for problem management
Escalate
Info exhausted and problem not resolved—Create trouble ticket
Act
Perform task that can produce more info or recovery
Evaluate
Get info from results of the actions

Figure 3: UCA Automation workflow

1.2 Prerequisites

The following are the prerequisites for implementing UCA Automation. Determine the following components before implementing.

- The domain or the service to be automated.
- For example, mobile services, MPLS, ADSL, LTE, ATM, and so on.
  - Identify the service for which the custom automation should be created.
  - All problems in that domain and the resolution mechanisms, including:
  - The problem scenario, the characteristics of the root problem, and the filter to be used to isolate this problem.
  - The specific problem/resolution tree for any of the root problems.
  - All resolution actions required for each sub-problem in the problem tree.
  - The input and output parameters for all actions.
  - The method of deducing the output parameters from the raw output using the regex/XML parser.
  - All possible outcomes for the actions.
  - Note whether the outcomes are binary or not.
  - Make a differentiation between the primary problems and the results of the actions.
- The decision tree to be built using these problems, actions, and outcomes.

1.3 Implementation

Use the following procedure to implement UCA Automation.

1. Create the domain or service to be automated.

   For example, mobile services, MPLS, ADSL, LTE, ATM, and so on.

   a. Create the service according to [R2] chapter 8.

2. Create all the possible problems in that domain and the resolution mechanisms, including the following:

   a. A UCA EBC PD value-pack depicting the identified problem scenario with appropriate filters and time-window according to [R3].

   b. Integrate this value-pack with UCA Automation Foundation value-pack as described in Chapter 2.

   c. All resolution actions identified to handle each problem and the outcome of actions. For more details, refer [R2] chapter 8.

   d. Appropriate input and output parameters for all actions.

   e. Select the appropriate output parameters and their respective parsers.

   f. Integrate with the UCA Automation as per Chapter 3.

3. Create a decision tree with these problems, actions, and outcomes according to the instructions in [R2] Chapter 8.

Chapter 2
Integrate with UCA Automation Foundation value pack

HPE UCA Automation Foundation value pack provides the capability to determine the next resolution action based on a reported problem.

The domain specific PD value pack determines and isolates the problem, and delegates the alarm object to the UCA Automation Foundation Value-Pack. After receiving the alarm object with appropriate problem qualification from the network specific PD value pack, the system searches for the resolution in the decision tree available in the Neo4J database and picks an appropriate action.

Based on the action, the foundation value-pack sends out the following XML request to UCA Automaton console for applying the resolution:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg xmlns="http://types.ws.ucaautomation.hp.com/">
  <header>
    <ActionRequest Originator="alarm" OpenLoop="true" Mode="demo">
      <ActionId>100</ActionId>
      <Operation>test_bsc_interface</Operation>
      <OrignatorId>operation_context uca_pbalarm alarm_object 4</OrignatorId>
      <SourceIdentifier>TeMIP EMS</SourceIdentifier>
      <OriginatingManagedEntity>osi_system site_site1</OriginatingManagedEntity>
      <Problem>bsc_interface_down</Problem>
      <ActionPreset>false</ActionPreset>
      <DispatchType>HPSA</DispatchType>
      <ActionExecutionMode>Asynchronous</ActionExecutionMode>
<ActionIdList>bsc_interface_down-&gt;test_bsc_interface[100:5]#</ActionIdList>
    </ActionRequest>
  </header>
  <body>
    <Parameters>
      <Parameter>
        <attribute> interface_ip_address</attribute>
        <value>192.168.1.2</value>
      </Parameter>
    </Parameters>
    <Service>
      <serviceTypeID>MobileServices</serviceTypeID>
      <serviceInstanceID></serviceInstanceID>
    </Service>
    <Resource>
      <resourceTypeID>C3600</resourceTypeID>
      <resourceInstanceID>C3600.ind.hpe.com|Ethernet1/0:C3620</resourceInstanceID>
    </Resource>
```

```
   </body>
</msg>
```

2.1 Integrate PD value pack with UCA Foundation pack

After you create the PD value pack with the appropriate problem scenario, the PD value pack provides a complete problem qualification to the UCA Foundation value pack. To provide the problem qualification, use the following procedure.

1. Set the values of the following attributes in the generated Problem Alarm using the calculateProblemAlarmOtherAttribute method.

Table 2: calculateProblemAlarmOtherAttribute description

Attribute	Description
Problem	A predefined Problem Name as defined in the decision tree. The format should be <ServiceType>:<Problem>
Resourceinstance	Resource instance based on how the resource is understood by the activation engine.
Evp	Evaluate Value Pack Name You have to create a network specific evaluate pack if the integrator wants to perform a complex integration and mechanism to determine the next possible problem. After receiving the response for an action, the Foundation value pack intercepts and delegates to a network specific evaluate value pack for further deduction of the problem.
Evpscenario	Evaluate Value Pack scenario The specific scenario to which the system delegates the response.

Shown below is a sample code describing the way to override the calculateProblemAlarmOtherAttribute method.

```
public void calculateProblemAlarmOtherAttribute(Group group, Action
action) throws Exception {

      if (LOG.isTraceEnabled()) {
            LogHelper.enter(LOG,
"Problem_Site.calculateProblemAlarmOtherAttribute()");
      }

      action.addCommand("Resourceinstance",
"C3600.ind.hpe.com|Ethernet1/0:C3600");
      action.addCommand("Evp","UCA_Automation_DomainExample_UCA_EV");
      action.addCommand("Evpscenario","evaluate");
      action.addCommand("Problem",
"MobileServices:bsc_interface_down");
```

```
      if (LOG.isTraceEnabled()) {
            LogHelper.exit(LOG,
"Problem_Site.calculateProblemAlarmOtherAttribute()",action.toFormatt
edString());
      }
}
```

The format of the “Resourceinstance” updated in the Problem Alarm must be as follows

<Resource Instance>:<Resource Type>

If the Resource Type is not applicable then specify the value as **None**. The format should be maintained in case the “Resouceinstance” is modified in the Evaluate value pack

The Resource type is optionally used by the workflow’s in HPE SA Network Specific value pack to parse diagnostic results using regular expressions.

The attributes “Evp” and “Evpscenario” are necessary only if responses from every action must be evaluated to decide the next action. If the “Evp” and “Evpscenario” are not specified the Foundation Value Pack uses the actionstatus (Passed or Failed) to retrieve the next action

2. Set or append the following text to the attribute “`userText`” in in the Problem Alarm “`to_be_processed_by_UCAAutomation`”

   This can be performed by overriding the calculateProblemAlarmUserText method

3. To provide a complete problem qualification to the UCA Foundation value pack, perform one of the following:

   a. (Optional) If the source of the alarms is TeMIP, configure the PD value pack to create the Problem Alarm in a separate Operation Context called `uca_pbalarm`.

   b. Configure it in the `ProblemXMLConfig.xml` file. Following is snippet of this file if the mediation is through UMB.

```
<actions>
    <defaultActionScriptReference>Exec_localhost</defaultActionScriptReference>
    <action name="TeMIP EMS">
        <actionReference>TeMIP_AO_Directives_localhost</actionReference>
        <actionClass> com.hp.uca.expert.vp.common.actions.temip.TeMIPActionsFactory </actionClass>
        <attributeUsedForKeyDuringRecognition>userText</attributeUsedForKeyDuringRecognition>
        <attributeUsedForKeyPbAlarmCreation>User_Text</attributeUsedForKeyPbAlarmCreation>
            <strings>
                <string key="ocName"><value>uca_pbalarm</value></string>
            </strings>
            …
            …
    </action>
</actions>
```

   c. If the source of alarms is not TeMIP, delegate the Problem alarm to the UCA Foundation value pack using the delegateEventToScenario() or applyOrchestration() API.

   d. For more details, refer to the UCA-EBC API documents.

   e. If the first action in the decision tree is a closed loop test and input parameters are necessary for executing this actions, the PD value pack can pass parameters by setting the custom attributes "`Parameternames`" and "`Parametervalues`" of Problem Alarm. The value must be a comma separated string.

```
public void whatToDoWhenProblemAlarmIsAttachedToGroup(Group group)throws Exception {
        super.whatToDoWhenProblemAlarmIsAttachedToGroup(group);
        Alarm problemAlarm = group.getProblemAlarm();
        problemAlarm.setCustomFieldValue("Parameternames", "interface_ip_address,param_1");
        problemAlarm.setCustomFieldValue("Parametervalues", "127.0.0.1,value_1");
        getScenario().applyOrchestration(problemAlarm);
}
```

o

2.1.1 NOM specific Configuration

If the mediation bus is NOM, for an alarm configure the action reference in ProblemXmlConfig.xml as follows.

```
<p1:action name="TeMIP EMS">
        <p1:actionReference>TeMIP_AO_Directives_localhostNOM</p1:actionReference>
        <p1:actionClass>com.hp.uca.expert.vp.common.actions.temip.TeMIPActionsFactory</p1:action
Class>
        <p1:attributeUsedForKeyDuringRecognition>userText</p1:attributeUsedForKeyDuringRecog
nition>
        <p1:attributeUsedForKeyPbAlarmCreation>User_Text</p1:attributeUsedForKeyPbAlarmCreat
ion>
        <p1:strings>
                <p1:string key="ocName">
                        <p1:value>uca_pbalarm</p1:value>
                </p1:string>
                ...
                ...
        <p1:strings>
</p1:action>
```

For a trouble ticket action, configure action reference as follows in ProblemXmlConfig.xml file.

```
<p1:troubleTicketAction name="TeMIP TT">
        <p1:actionReference>TeMIP_TT_Directives_localhostNOM</p1:actionReference>
        <p1:actionClass>com.hp.uca.expert.vp.pd.actions.TeMIPTroubleTicketActionsFactory</p1:acti
onClass>
        <p1:strings>
                ...
                ...
        </p1:string>
</p1:troubleTicketAction>
```

In ValuepackConfiguration.xml file, have only <mediationFlow/> elements.

```
<mediationFlow name="temipFlow" actionReference="TeMIP_FlowManagement"
flowNameKey="flowName"  lastEventReceivedFirstDuringResynchronization="true">
        <!-- Comment out the flowCreation and flowDeletion sections to use static flows instead of
dynamic flows -->
        <flowCreation>
                <actionParameter>
                        <key>operation</key>
                        <value>CreateFlow</value>
                </actionParameter>
                <actionParameter>
                        <key>flowType</key>
                        <value>dynamic</value><!-- flowType can only be dynamic in the case of
flowCreation -->
                </actionParameter>
                <actionParameter>
                        <key>operationContext</key>
                        <value>uca_network</value>
                </actionParameter>
                <actionParameter>
                        <key>operationContext</key>
                        <value>uca_pbalarm</value>
                </actionParameter>
        </flowCreation>
        <flowDeletion>
                <actionParameter>
                        <key>operation</key>
                        <value>DeleteFlow</value>
                </actionParameter>
                <actionParameter>
                        <key>flowType</key>
                        <value>dynamic</value><!-- flowType can only be dynamic in the case of
flowDeletion -->
                </actionParameter>
        </flowDeletion>
        <flowResynchronization>
                <actionParameter>
                        <key>operation</key>
                        <value>ResynchFlow</value>
                </actionParameter>
                <actionParameter>
                        <key>flowType</key>
                        <value>dynamic</value><!-- flowType can be either static or dynamic -->
                </actionParameter>
        </flowResynchronization>
        <flowStatus>
                <actionParameter>
                        <key>operation</key>
                        <value>StatusFlow</value>
                </actionParameter>
                <actionParameter>
```

```
                                <key>flowType</key>
                                <value>dynamic</value><!-- flowType can be either static or dynamic -->
                        </actionParameter>
                </flowStatus>
</mediationFlow>
```

2.1.2 UMB specific Configuration

If the mediation bus is UMB, configure the action reference in ProblemXmlConfig.xml as follows.

```
<p1:action name="TeMIP EMS">
        <p1:actionReference>TeMIP_AO_Directives_localhost</p1:actionReference>
        <p1:actionClass>com.hp.uca.expert.vp.common.actions.temip.TeMIPActionsFactory</p1:action
Class>
        <p1:attributeUsedForKeyDuringRecognition>userText</p1:attributeUsedForKeyDuringRecog
nition>
        <p1:attributeUsedForKeyPbAlarmCreation>User_Text</p1:attributeUsedForKeyPbAlarmCreat
ion>
        <p1:strings>
                <p1:string key="ocName">
                        <p1:value>uca_pbalarm</p1:value>
                </p1:string>
        </p1:strings>
        ...
        ...
</p1:action>
```

For a trouble ticket action, configure action reference as follows in ProblemXmlConfig.xml file.

```
<p1:troubleTicketAction name="TeMIP TT">
        <p1:actionReference>TeMIP_TT_Directives_localhost</p1:actionReference>
        <p1:actionClass>com.hp.uca.expert.vp.pd.actions.TeMIPTroubleTicketActionsFactory</p1:acti
onClass>
        <p1:strings>
                ...
                ...
        </p1:strings>
</p1:troubleTicketAction>
```

In ValuepackConfiguration.xml file, have only <UMBmediationFlow/> elements.

```
<mediationFlows>
        <UMBmediationFlow name="temipFlow" automaticStart="true"
targetFlowName="UCAAutomationTeMIPFlow" targetAdapterName="TeMIP">
                <flowParameters>
                        <flowParameter value="uca_network" key="operationContext"/>
                        <flowParameter value="uca_pbalarm" key="operationContext"/>
                </flowParameters>
        </UMBmediationFlow>
```

```
</mediationFlows>
```

2.2 Integrate Evaluate value pack with UCA Foundation pack

Creating and integrating the network specific evaluate value pack is optional.

Create and integrate the network specific evaluate value pack when you want to interpret the resultant output in very specific ways other than a test passed or test failed criterion.

You can use this value pack to analyze the output from the previous action and can determine the next step or action outcome to be passed to the foundation value pack. This value pack can contain several scenarios to interpret different outputs from different PD scenarios. It can also contain 1 * n relationships between number of domain specific PD value packs, which represent one scenario each, and evaluate value pack, which represents n scenarios.

You should have EBC rules skill to write this value pack. The following snippet shows a scenario where an action response with some parameters is intercepted, the action outcome is deduced, and alarm attributes updated in TeMIP are picked up by the foundation value pack for further processing. The output parameters are in the following format

```
<Parameters xmlns=""http://types.ws.ucaautomation.hp.com/"">
  <Parameter>
    <attribute>packet_loss</attribute>
    <value>100</value>
    <type>String</type>
  </Parameter>
  <Parameter>
    <attribute>packet_sent</attribute>
    <value>4</value>
    <type>String</type>
  </Parameter>
</Parameters>
```

The Evaluate Value pack must send an Attribute Value Change to the UCA Automation Foundation Value Pack with all the modified custom attributes. It must contain an attribute with name “avcfromevp” and value true. This is a go ahead from the Evaluate VP that the Foundation Value Pack can continue traversing the decision tree. The value pack must persist the modified value of the custom attribute “actionidlist” in the NMS.

```
<AlarmAttributeValueChangeInterface>
        <identifier>operation_context cls.eby.oc.four_g_oc alarm_object 31</identifier>
        <sourceIdentifier>TeMIP EMS</sourceIdentifier>
        <originatingManagedEntity>trail_01_osilocal</originatingManagedEntity>
        <alarmType>COMMUNICATIONS_ALARM</alarmType>
        <probableCause>Fire</probableCause>
        <perceivedSeverity>MINOR</perceivedSeverity>
```

```
            <attributeChanges>
                        <attributeChange name="avcfromevp" newValue="true" oldValue="" />
                        <attributeChange name="parameternames" newValue="tt_id,alarm_handled_time"
oldValue="" />
                        <attributeChange name="parametervalues" newValue="IM1923845,1457426611000"
oldValue="" />
                        <attributeChange name="actionidlist" newValue="bsc_interface_down-
>test_bsc_interface[148:111]#test_bsc_interface_PASSED" oldValue="" />
            </attributeChanges>
</AlarmAttributeValueChangeInterface>
```

```
rule "Evaluate Action response Rule"
no-loop
 when
            $alarm : EvaluateActionResponse(justInserted == true)
 then
            LogHelper.enter(theScenario.getLogger(), drools.getRule().getName());

            theScenario.getLogger().trace(Messages.EVALUATE_ACTION_RESPONSE_RULE_HAS_FIRE
D_CORRECTLY.getMessage(new Object[]{$alarm.toFormattedString()}));
            $alarm.setJustInserted(false);
            $alarm.setScenario(theScenario);
            $alarm.evaluateOutputParams();

            theScenario.getLogger().info(Messages.RETRACTING_THE_ALARM_FROM_EVALUATE_AC
TION_RESPONSE_RULE.getMessage());
            theScenario.getSession().retract($alarm);
            LogHelper.exit(theScenario.getLogger(), drools.getRule().getName());
end
```

```
AlarmAttributeValueChange attributeValueChange;
AttributeChanges attrChanges;
private String actionIdList;

public void evaluateOutputParams()
{
   constructAVC();
      if (this.getCustomFieldValue("action")
!= null && this.getCustomFieldValue("problem")
!= null &&
this.getCustomFieldValue("problem").contains("MobileServices"))

{

if (this.getCustomFieldValue("outputparameters") != "") {
      String actionOutParams =
this.getCustomFieldValue("outputparameters");

      try {
      this.getDomainProblemInfo();
      this.getActionIdListfromAlarm();
if (actionOutParams.contains("packet_loss")) {

      this.evaluatePacketLossParam(actionOutParams);

}
 else if (actionOutParams.contains("available_interface_name"))

{
      this.evaluateAvailInterfParam(actionOutParams);
                        }
else if(actionOutParams.contains("tt_id")){
      this.evaluateTTParam(actionOutParams);
} else {
{
      LOG.info("Alarm outputParameters doesn't have the required
output parameters to process");

                  }
}
 catch (Exception e)
{
LOG.error("Exception occurred : " + e.getMessage());

      this.getScenario().setStatus("Exception occurred:
"+e.getMessage(),  ScenarioStatus.Degraded);
                  }
            }
else if
(this.getCustomFieldValue("outputparameters") == "")
{

constructDefaultActionOutcome();

LogHelper.method(LOG,
      "EvaluateActionResponse.evaluateOutputParams()",
      " Alarm is enriched with new Action Outcome name based on the
action and actionStatus");
            }
      }

//add the modified actionidlist and actionstatus in the AVC
//set to Foundation Value Pack
enrichAVC("actionidlist",this.getCustomFieldValue("actionidlist");
enrichAVC("actionstatus",this.getCustomFieldValue("actionstatus");

getScenario().applyOrchestration(attributeValueChange);
}

public void getActionIdListfromAlarm() {
      String actionIdList =
this.getCustomFieldValue("actionidlist");
      setActionIdList(actionIdList);
}

public void setActionIdList(String actionIdList) {
      this.actionIdList = actionIdList;
}
```

```
public void evaluatePacketLossParam(String actionOutParams) throws
Exception {
      //update the actionidlist and actionstatus in the Temip
   //parse the outputparameters xml to get packet_loss
      ParametersType parameters=
parseActionOutputParameters(actionOutParams);
//from the outputparameters get the packet loss
      ParameterType actionParameter=
getActionParameter(parameters.getParameter(), “packet_loss”);

      String packetLossValue = actionParameter.getValue();
      //In temip the custom attribute actionidlist is Actionidlist,
but in the working memory it is actionidlist
      //In temip the custom attribute actionstatus is Actionstatus,
but in the working memory it is actionstatus
      if (packetLossValue == null ||
packetLossValue.equalsIgnoreCase("null")
      || packetLossValue.equals("")) {

updateAlarmCustomAttr("Actionidlist", "actionidlist",
actionIdList+"test_bsc_interface_failed");

updateAlarmCustomAttr("Actionstatus", "actionstatus", "FAILED");
      }
else if
(Integer.parseInt(packetLossValue) > 60)
{
      //Packetloss is greater than 60, override the actionstatus from
passed to failed

updateAlarmCustomAttr("Actionidlist", "actionidlist",
actionIdList+"test_bsc_interface_failed");

if
(this.getCustomFieldValue("actionstatus").equalsIgnoreCase("PASSED"))
{
      updateAlarmCustomAttr("Actionstatus", "actionstatus",
"FAILED");

      }
      enrichAVC("serviceinstance",
this.getCustomFieldValue("serviceinstance"));
      } else
{
      //Packetloss is less than 60. override the actionstatus from
failed to passed

updateAlarmCustomAttr("Actionidlist", "actionidlist",
actionIdList+"test_bsc_interface_passed");

      if
(!this.getCustomFieldValue("actionstatus").equalsIgnoreCase("PASSED")
) {
      updateAlarmCustomAttr("Actionstatus", "actionstatus",
"FAILED");
            }
      }

      private void constructAVC() {
            attributeValueChange = new AlarmAttributeValueChange();

            attrChanges = new AttributeChanges();
            AttributeChange = new AttributeChange();
            attributeChange.setName("avcfromevp");
            attributeChange.setNewValue("true");
            attributeChange.setOldValue("");
            attrChanges.getAttributeChange().add(attributeChange);

            attributeValueChange.setAttributeChanges(attrChanges);
            attributeValueChange.setIdentifier(this.getIdentifier());

      attributeValueChange.setSourceIdentifier(this.getSourceIdentifi
er());
            attributeValueChange.setOriginatingManagedEntity(this
                        .getOriginatingManagedEntity());

      attributeValueChange.setOriginatingManagedEntityStructure(this
                        .getOriginatingManagedEntityStructure());
            attributeValueChange.setAlarmType(this.getAlarmType());

      attributeValueChange.setProbableCause(this.getProbableCause());

      attributeValueChange.setPerceivedSeverity(this.getPerceivedSeve
rity());

      }

      public void enrichAVC(String attributeName, String
attributeValue) {
            AttributeChange = new AttributeChange();
            attributeChange.setName(attributeName);
            attributeChange.setNewValue(attributeValue);
            attributeChange.setOldValue("");
            attrChanges.getAttributeChange().add(attributeChange);
      }

      private void constructDefaultActionOutcome() {

            String actionStatus = this
      .getCustomFieldValue("actionstatus");
            String action = this.getCustomFieldValue("action");
            String actionOutcome = action + "_" + actionStatus;
            String actionIdLists = this
      .getCustomFieldValue("actionidlist");
            this.updateAlarmCustomAttr("Actionidlist",
                        "actionidlist", actionIdLists +
actionOutcome);
            actionIdList = actionIdLists + actionOutcome;
      }

public void updateAlarmCustomAttr(String customFieldName,
      String aoFieldName, String newValue) {
      Action = null;

      String message = "updateAlarmCustomAttr " + customFieldName + "
for " + this.getIdentifier();
      try {
            synchronized (this) {
            action = new Action("TeMIP_AO_Directives_localhost");
      action.addCommand(AODirectiveKey.DIRECTIVE_NAME,
                                  AODirective.SET);
      action.addCommand(AODirectiveKey.ENTITY_NAME,
                                  this.getIdentifier());
      action.addCommand(customFieldName, newValue);
      action.executeSync();

      if (action.getActionStatus().equals(ActionStatus.Completed)) {
                            this.setCustomFieldValue(aoFieldName,
newValue);
      } else {
            LOG.info("Exception occured: Unable to update the alarm
custom attribute in TeMIP:"+ customFieldName);

      }
      }
      } catch (Exception e) {
                  LogHelper.logErrorDebug(LOG,
                              "Exception while initializing Action: "
+ message, e);

            }
      }

public ParametersType parseActionOutputParameters(String
outputParameters){
   ParametersType parametersType=null;
   try {
      JAXBContext jaxbContext = JAXBContext

      .newInstance(com.hp.ucaautomation.ws.types.ParametersType.class
);
      Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();

      JAXBElement<ParametersType> warppedHeader =
(JAXBElement<ParametersType>) unmarshaller
                        .unmarshal(new StreamSource(new
StringReader(outputParameters)),

      com.hp.ucaautomation.ws.types.ParametersType.class);
      parametersType = warppedHeader.getValue();

   } catch (JAXBException e) {

   }
   return parametersType;
}

public void ParameterType getActionParameter(List<ParameterType>
responseParameters, String paramName){
   ParameterType responseParameterType = null;
   if (responseParameters != null) {
      Iterator<ParameterType> responseParametersIterator =
responseParameters
                        .iterator();

      while (responseParametersIterator.hasNext()) {
         responseParameterType = (ParameterType)
responseParametersIterator
                              .next();
```

```
if(responseParameterType.getAttribute().equalsIgnoreCase(paramName)){
            break;
         }

      }

   }
   return responseParameterType;
}
}
```

2.2.1 NOM specific configuration

If the mediation bus is NOM, configure the action reference in UCAEvaluate.properties as follows.

```
#UMB
#actionReference=TeMIP_AO_Directives_localhost
#NOM
actionReference=TeMIP_AO_Directives_localhostNOM
```

In ValuepackConfiguration.xml file, have only <mediationFlow/> elements .

```
<mediationFlow name="temipFlow" lastEventReceivedFirstDuringResynchronization="true"
flowNameKey="flowName" actionReference="TeMIP_FlowManagement">
        <flowResynchronization>
                <actionParameter>
                        <key>operation</key>
                        <value>ResynchFlow</value>
                </actionParameter>
                <actionParameter>
                        <key>flowType</key>
                        <value>dynamic</value>
                        <!-- flowType can be either static or dynamic -->
                </actionParameter>
        </flowResynchronization>
        <flowStatus>
                <actionParameter>
                        <key>operation</key>
                        <value>StatusFlow</value>
                </actionParameter>
                <actionParameter>
                        <key>flowType</key>
                        <value>dynamic</value>
                        <!-- flowType can be either static or dynamic -->
                </actionParameter>
        </flowStatus>
</mediationFlow>
```

2.2.2 UMB Specific Configuration

If the mediation bus is UMB, configure the action reference in UCAEvaluate.properties as follows.

```
#UMB
actionReference=TeMIP_AO_Directives_localhost
#NOM
#actionReference=TeMIP_AO_Directives_localhostNOM
```

In ValuepackConfiguration.xml file, have only <UMBmediationFlow/> elements.

```
<mediationFlows>
        <UMBmediationFlow name="temipFlow" automaticStart="true"
targetFlowName="UCAAutomationTeMIPFlow" targetAdapterName="TeMIP">
                <flowParameters>
                        <flowParameter value="uca_pbalarm" key="operationContext"/>
                </flowParameters>
        </UMBmediationFlow>
</mediationFlows>
```

2.2.3 UCA-EBC configurations

For the Problem Alarm to be routed between the Foundation and the Evaluate value pack, Orchestration Routes have to be defined in the OrchestraConfiguration.xml file of the UCA-EBC server instance, in the `${UCA_EBC_INSTANCE}/conf` folder. This file is only loaded at UCA-EBC server instance start (static loading).

```
<Routes>
        <Route name="Copy from  from UCA Automation Foundation VP to  from UCA Automation
EV VP ">
                <COPY>
                        <Source>
                                <ValuePackNameVersion>UCA_Automation_Foundation_UCA-
V2.1-1A</ValuePackNameVersion>

        <ScenarioName>UCA_Automation_Foundation_UCA.requestresponse</ScenarioName>
                        </Source>
                        <Destinations>
                                <Destination>
                                        <Target>

        <ValuePackNameVersion>UCA_Automation_DomainExample_UCA_EV-V2.1-
1A</ValuePackNameVersion>

        <ScenarioName>UCA_Automation_DomainExample_UCA_EV.evaluate</ScenarioName>
                                        </Target>
                                </Destination>
                        </Destinations>
                </COPY>
        </Route>
```

```
            <Route name="Copy from UCA Automation EVP to from UCA Automation Foundation VP ">
                        <COPY>
                                    <Source>

            <ValuePackNameVersion>UCA_Automation_DomainExample_UCA_EV-V2.1-
1A</ValuePackNameVersion>

            <ScenarioName>UCA_Automation_DomainExample_UCA_EV.evaluate</ScenarioName>
                                    </Source>
                                    <Destinations>
                                                <Destination>
                                                            <Target>

            <ValuePackNameVersion>UCA_Automation_Foundation_UCA-V2.1-
1A</ValuePackNameVersion>

            <ScenarioName>UCA_Automation_Foundation_UCA.requestresponse</ScenarioName>
                                                            </Target>
                                                </Destination>
                                    </Destinations>
                        </COPY>
            </Route>
</Routes>
```

The Problem Alarm is cascaded from Foundation to the Evaluate value pack. The evaluate value pack processes the result of the previous action and determines the next action to be executed and notifies the Foundation value pack via an Attribute Value Change. Hence looping must be allowed by explicitly setting to true the `uca.ebc.orchestra.loops.allowed` property from the `uca-ebc.properties` file (found in the `${UCA_EBC_INSTANCE}/conf` directory).

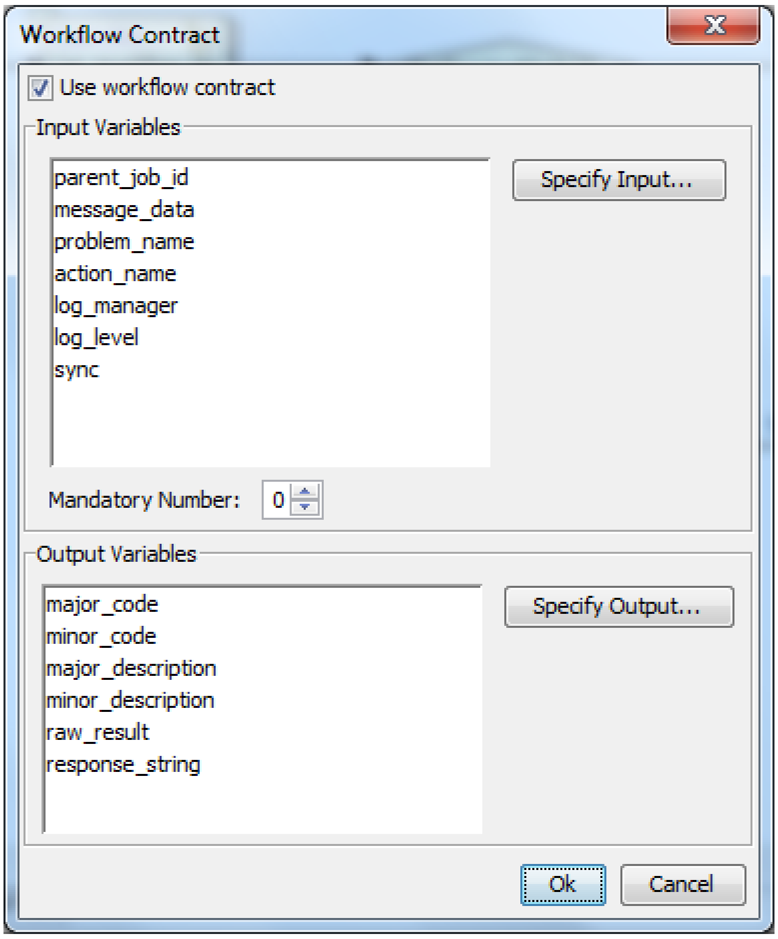
Chapter 3
Integrate with HPE SA UCA Automation Controller

The HPE SA framework handles the how part of the resolution action, which is required for developing new value packs for custom automation. To have an integrated view, the UCA Automation provides a controller workflow with which all the domain specific workflows are integrated.

The task request with the dispatch type as **HPSA** from UCA Automation Console invokes the UCA Controller workflow of the HPE SA Foundation Value Pack. Hence, the point of entry for the task request and point of exit for the task response is the UCA Controller workflow. All domain specific workflows are invoked from this workflow.

1. Select **UCA/Parameter** -> **Workflow Templates** view in the HPE SA inventory.
2. Create the mapping to the child domain specific workflows.

   Create a mapping of a combination of ServiceType and ActionName with the child domain specific workflow, which is designed to handle such scenarios. The HPE SA workflow node used to invoke the domain workflow can be either StartJobAndWait or ExecuteMacro. If ExecuteMacro is specified then a contract must be defined in the domain specific workflow as follows. The contract can be defined in the Workflow Designer using the Workflow -> Workflow Contract menu. A Workflow Contract is used to define input and output parameters for a given workflow



[✓] Use workflow contract

Input Variables:

```
parent_job_id
message_data
problem_name
action_name
log_manager
log_level
sync
```

Mandatory Number: 0

Output Variables:

```
major_code
minor_code
major_description
minor_description
raw_result
response_string
```

Figure 4: UCAController conract

3.1 Workflow of a task request

When a task request from the UCA Automation Console invokes the UCA Controller workflow, the Workflow Template is searched automatically to fetch the corresponding child workflow based on the ServiceType and ActionName provided in the task request XML message.

The following snippet shows the Task Request message.

```
<m:msg xmlns:m="http://types.ws.ucaautomation.hp.com/">
        <m:header>
                <m:TaskRequest Mode="real" OpenLoop="true" Originator="alarm">
                        <m:ActionId>100</m:ActionId>
                        <m:ActionName>test_bsc_interface</m:ActionName>
                        <m:ActionType>test</m:ActionType>
                        <m:Operation>Start</m:Operation>
                        <m:TaskId>110</m:TaskId>
                        <m:OriginatorId>operation_context uca_pbalarm alarm_object
4</m:OriginatorId>
                        <m:Problem>bsc_interface_down</m:Problem>
                </m:TaskRequest>
        </m:header>
        <m:body>
                <m:Parameters>
                        <m:Parameter>
                                <m:attribute>interface_ip_address</m:attribute>
                                <m:value>10.20.30.40</m:value>
                        </m:Parameter>
                </m:Parameters>
                <m:Service>
                        <m:serviceTypeID>MobileServices</m:serviceTypeID>
                        <m:serviceInstanceID/>
                </m:Service>
                <m:Resource>
                        <m:resourceTypeID>C3600</m:resourceTypeID>
                <m:resourceInstanceID>C3600.ind.hpe.com|Ethernet1/0</m:resourceInstanceID>
                </m:Resource>
        </m:body>
</m:msg>
```

NOTE: You should have controllers for each domain as shown in the Multi Domain Solution

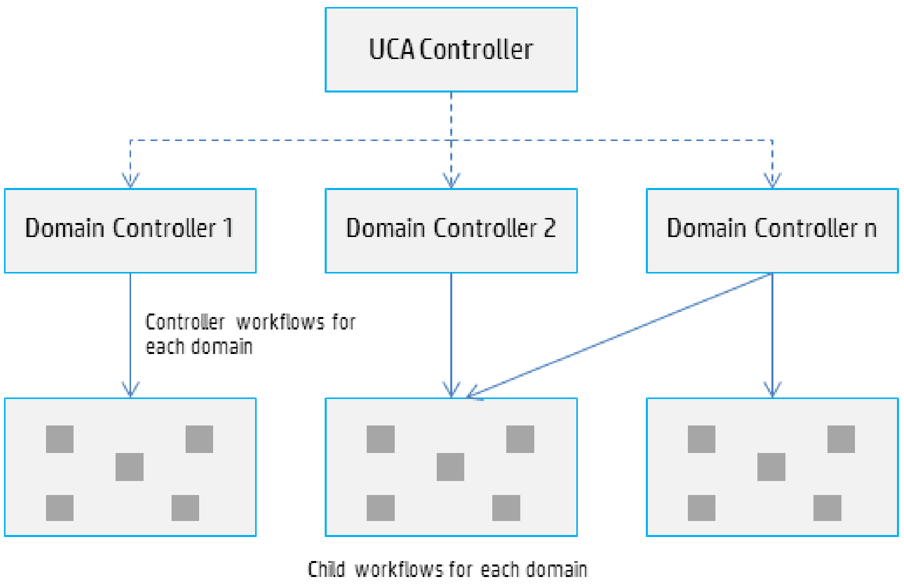


Figure 5: Multi Domain Solution

The following table shows the list of parameters parsed by the UCAController workflow when the domain workflow is invoked. It also shows the expected output case packet variables from the child workflow.

Table 3: UCAController parameters

Parameter	Input/Output	Description
JOB_ID	Input	Job id of the UCAController. The destination case-packet in the domain workflow is parent_job_id of type Integer
message_data	Input	Task request message received from the UCA Automation Console.
problem_name	Input	Name of the problem.
action_name	Input	Name of the diagnostic action.
log_manager	Input	Reference to the Log Manager configured in mwfm.xml. The default value is log_manager
log_level	Input	The Log level. The default value is INFORMATIVE
sync	Input	Flag indicating that the domain workflow must sync back to UCAController. The destination case-packet in the domain workflow is sync of type Boolean
major_code	Output	Status code for running the child workflow.
minor_code	Output	Status code with information on the execution status of the child workflows.
major_description	Output	Status message of the execution of the child workflow.
minor_description	Output	Status message with information on the execution of the child workflows.
Diagnostics	Output	The raw result of the application of the action.
response_string	Output	The output parameter and parsed values are concatenated in a format defined by UCA-EBC. These values are sent as a value in the `outputparameters` tag of the response to UCA–EBC. The format of the string is `<Parameters><Parameter><attribute>attribute1</attribute><value>value1</value><type>attribute1 type</type></Parameter><Parameter><attribute>attribute2</attribute><value>value2</value><type>attribute2 type</type></Parameter>...</Parameters>`

The following snippet shows the Task Response message.

```
<resp_msg xmlns="http://types.ws.ucaautomation.hp.com/">
        <header>
                <ActionId>100</ActionId>
                <TaskId>100</TaskId>
                <ActionInstanceId>1</ActionInstanceId>
                <OrignatorId>operation_context uca_pbalarm alarm_object 4</OrignatorId>
        </header>
        <body>
                <TaskResponse>
                        <MajorCode>
                                <Code>501</Code>
                                <Description>The test execution failed
      </Description>
                        </MajorCode>
```

```
                              <MinorCode>
                                        <Code/>
                                        <Description/>
                              </MinorCode>
                              <TaskStatus>FAILED</TaskStatus>
                              <Diagnostics>PING 10.20.30.40 (10.20.30.40) 56(84) bytes of data.
             --- 10.20.30.40 ping statistics
             --- 5 packets transmitted, 0 received, 100% packet loss, time 13999ms
                </Diagnostics>
                              <Parameters>
                                        <Parameter>
                                                  <attribute>packet_loss</attribute>
                                                  <value>100</value>
                                                  <type>String</type>
                                        </Parameter>
                              </Parameters>
                    </TaskResponse>
          </body>
</resp_msg>
```

Following is a snippet of the HPE SA Domain Workflow synchronizing with the UCAController.

```
<End-Handler>
          <Name>SyncHandler</Name>
          <Class-Name>com.hp.ov.activator.mwfm.component.builtin.SyncHandler</Class-Name>
          <Param name="job_id" value="parent_job_id"/>
          <Param name="queue" value="constant:controller_queue"/>
          <Param name="destination0" value="major_code"/>
          <Param name="variable0" value="major_code"/>
          <Param name="destination1" value="minor_code"/>
          <Param name="variable1" value="minor_code"/>
          <Param name="destination2" value="major_description"/>
          <Param name="variable2" value="major_description"/>
          <Param name="destination3" value="minor_description"/>
          <Param name="variable3" value="minor_description"/>
          <Param name="destination4" value="diagnostics"/>
          <Param name="variable4" value="raw_result"/>
          <Param name="variable5" value="response_string"/>
</End-Handler>
```

3.1.1 Status codes

Two sets of status codes are used when implementing domain value packs in HPE SA. The status code bundles are located at `${SOLUTION_ETC}/etc/config/messages` in the HPE SA Foundation value pack.

Major code—The major code gives high level information on the execution status and drives the state engine in the UCA Automation Console.

A sample of major codes and description in messages.properties file is as follows.

```
200=The test was successfully executed
201=The test was partially executed
210=Workflow execution success
300=Request received
400=Bad request, syntax error
401=Invalid request
500=Internal error
501=The test execution failed
```

Minor code—The minor code and description are secondary codes, which give more information on the status of the execution.

A sample of minor codes and description in messages.properties file is as follows.

```
402=Parameter: {0} cannot be null/empty
403={0}: {1} was not found in inventory
510={0}: {1} Not Found
511={0} has exceeded the threshold value {1}
512=Free {0} is not available
```

Follow the major code standard according to the HPE SA Foundation value pack, as it drives the state engine in the UCA Automation Console.

You should maintain the major and minor code message bundles in a similar way. The domain specific major code message bundle contains all the codes defined in the HPE SA Foundation value pack. The minor code message bundle can be defined as per the requirement.

3.1.2 Support for internationalization

The message bundles support internationalization with the help of the custom node. The ResourceBundleReader custom node is available with the Foundation value pack.

Change the file name of the bundle according to the national standards. For example, for French regional setting, the file name is `message_fr.properties`. By default the `message.properties` bundle is picked by the node.

Table 4: Parameters of the ResourceBundleReader node

Parameters	Input/Output	Description
bundle_path	Input	The path to the message bundle. In the foundation value pack, the bundle_path to the major code messages is `%SOLUTION_ETC%/config/messages/majorcodes`
resource_label	Input	Label of the message bundle. The label in the foundation value pack is messages.
Key	Input	Key in the resource bundle. Set the key to `500` if you want the description for this major code.
output_var	Output	Variable in which the fetched string should be stored.
param0...n	Input	This value replaces the constant/variable in the string fetched from the message bundle. `param0` replaces the occurrence of `{0}` in the text. `param1` replaces the occurrence of `{1}` in the text

3.2 Using HPE SA UCA Automation parser

The parser workflow provides a framework for parsing the diagnostic raw result received from the network resources after applying an action.

Both regular expression and XPath based parsing are supported.

1. Define the parser type when defining an action as the UCA Automation inventory.
2. Create the following directory structure in the solution where `<element_type>` is various types of the device.

```
${SOLUTION_ETC}/config/parser/regex/<elementype>/test_bsc_interface/parser.properties
```

This structure is an example for parsing the output result of a PING action using the regular expression parser.

The parsing information is maintained in the properties files in the `${SOLUTION_ETC}` directory. The properties file contains the mapping of the expected output parameters defined in the inventory for each diagnostic action to its respective regular expression or Xpath expression.

A sample of the `parser.properties` file is as follows.

```
#REGEX mapping for Action: execute_test_on_bsc
#DOMAIN_NAME = com.hp.ov.ucaautomation  (Constant)
#Key -- > DOMAIN_NAME + "." + <ACTION_NAME> + "." + <PARAMETER>
#ACTION_NAME corresponds to the ACTION_ID of AUTOMATION_ACTION table in inventory
#Each ACTION_ID has a list of PARAMETERS in the PARAMETERS table in inventory
#
#All '\' characters in the regex must be escaped for JAVA
```

```
#e.g regex pattern for  packetloss
#                    Lost\s=\s\d*\s\((\d*%)\sloss\)  ---- > Lost\s=\s\d*\s\((\d*%)\sloss\)
#group id is used to return the input subsequence captured during the match operation
#Key for group id -- > DOMAIN_NAME + "." + <ACTION_NAME> + "." + <PARAMETER> + "." + groupid

com.hp.ucaautomation.test_bsc_interface.packet_loss = (\d*)%\spacket loss,
com.hp.ucaautomation.test_bsc_interface.packet_loss.groupid = 1
com.hp.ucaautomation.test_bsc_interface.min_time = \s*Minimum\s=\s(\d*\w*)
com.hp.ucaautomation.test_bsc_interface.min_time.groupid = 1
```

3. Enter the values for the following parameters when invoking the Parser workflow.

Table 5: Parser parameters

Parameter	Input/Output	Description
parser_bundle_label	Input	The name of the parser bundle. In the example in Step 2, the bundle name is `parser`.
parser_bundle_path	Input	The path where parser bundles are available. In the example, the path is `${SOLUTION_ETC}/config/parser/regex/<elementype>/test_bsc_interface`.
parser_type	Input	The type of the parser (regex or xpath). In the previous, the value is `regex`.
action_name	Input	Name of the diagnostic action defined in inventory. In the example, the action name is `test_bsc_interface`.
raw_result	Input	This parameter is the case packet variable which contains the raw information. The raw information is parsed and the data is extracted.
message_data	Input	The request message that was received from the UCA Automation Console.
parameter_map	Output	This map variable contains the mapping of each output parameter of the action to its corresponding parsed values.
minor_code	Output	Status of the workflow execution. A value of `210` represents a successful execution.
minor_description	Output	Diagnostic information of the workflow execution.
response_string	Output	The output parameter and parsed values are concatenated in a format defined by UCA-EBC. This value is sent as a value in the `outputparameters` tag of the response to UCA–EBC. The format of the string is `<Parameters><Parameter><attribute>attribute1<`

		`/attribute><value>value1</value><type>attribute1 type</type></Parameter><Parameter><attribute>attribute2</attribute><value>value2</value><type>attribute2 type</type></Parameter>...</Parameters>`

Chapter 4
Advanced Integration features

4.1 Mapping multiple scenarios to a generic scenario

The mappers' configuration file can be used to map several problem alarms (having the same problem resolution flow in the decision tree) to a more generic problem

The mapping is specified in `mappers.xml` file in the `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/requestresponse` directory

```
<?xml version="1.0" encoding="UTF-8" ?>
<mappers xmlns="http://hp.com/uca/expert/instancemapper">
        <mapper name="environmental_process" >
        <extract>
                <fieldName>problem</fieldName>
                <matcher>(.*)Library_Eltek_Fuse_Fail| (.*)Library_Smoke_Alarm</matcher>
                <mappedTo>environmental_default</mappedTo>
        </extract>
</mapper>
<mapper name="RAN" >
        <extract>
                <fieldName>problem</fieldName>
                <matcher>(.*):Library_HW_SCTPLinkFault|(.*):Library_NSN_TransmissionAlarm
</matcher>
                <mappedTo>ran_default</mappedTo>
        </extract>
        <extract>
                <fieldName>problem</fieldName>
                <matcher>(.*):Library_HW_ACMainsFail|(.*):Library_NSN_ACMainsFail</matcher>
                <mappedTo>ac_mains_failure_default</mappedTo>
        </extract>
</mapper>
</mappers>
```

In the example above the name attribute of the mapper tag defines the service type (domain) in the UCA Automation's decision tree. The <matcher> Tag defines the regular expression to be applied to the problem field of the alarm. If the regular expression match succeeds the problem alarm is mapped to the more generic problem alarm name i.e. `environmental_default`. This allows the integrator to club multiple problem alarms and have a single default alarm, which maps to a branch in the decision tree (instead of duplicating the decision tree branch for each problem alarm). The configuration file can have several mappers as shown.

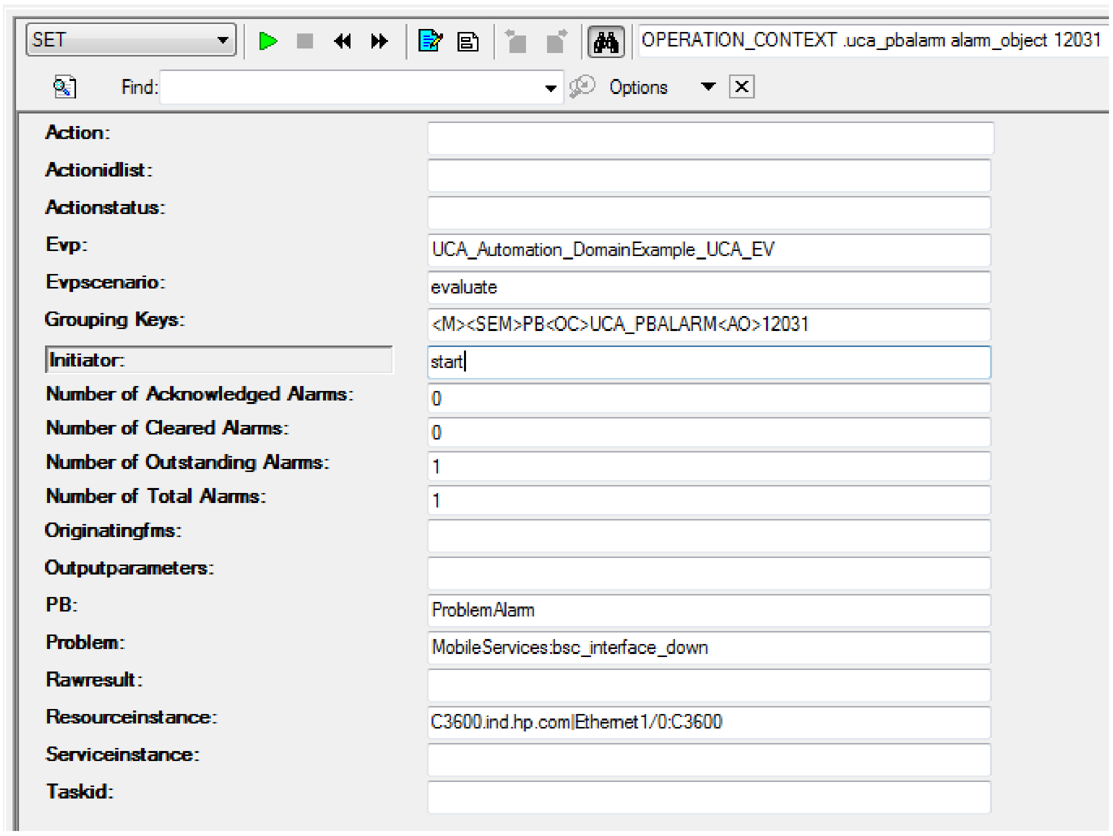
Having such a mapping is optional and should be only used when there are several problem alarms in a domain (service type) having the same problem resolution path in the decision tree. If there is no mapping specified in the mappers configuration file for a particular problem, then the actual problem name and the service type is used to look up in the decision tree.

4.2 Scenario based Automation

The default behavior of UCA Automation is to trigger the resolution process automatically without any user intervention. This can be controlled by a flag defined in `UCAAutomation.properties` file in the `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf` directory

```
manual_mode=false
```

If this is set to true, the automation process is not triggered till the user sets the value of the custom attribute Initiator to "`start`" in the Problem Alarm. This can be performed either from the TeMIP client or sending an Attribute Value Change message



OPERATION_CONTEXT .uca_pbalarm alarm_object 12031

Action:	
Actionidlist:	
Actionstatus:	
Evp:	UCA_Automation_DomainExample_UCA_EV
Evpscenario:	evaluate
Grouping Keys:	<M><SEM>PB<OC>UCA_PBALARM<AO>12031
Initiator:	start
Number of Acknowledged Alarms:	0
Number of Cleared Alarms:	0
Number of Outstanding Alarms:	1
Number of Total Alarms:	1
Originatingfms:	
Outputparameters:	
PB:	ProblemAlarm
Problem:	MobileServices:bsc_interface_down
Rawresult:	
Resourceinstance:	C3600.ind.hp.com|Ethernet1/0:C3600
Serviceinstance:	
Taskid:	

Figure 6: Automation on Demand

Here is an example of sending an AVC message to start by using the UCA-EBC alarm injector utility

```
<?xml version="1.0" encoding="UTF-8"?>
<alarms:Alarms xmlns:alarms="http://hp.com/uca/expert/x733Alarm">
        <alarms:AlarmAttributeValueChangeInterface>
        <alarms:identifier>operation_context uca_pbalarm alarm_object 12031</alarms:identifier>
        <alarms:sourceIdentifier>user</alarms:sourceIdentifier>
        <alarms:originatingManagedEntity>osi_system site_site1</alarms:originatingManagedEntity>
```

```
            <alarms:additionalText></alarms:additionalText>
            <alarms:attributeChanges>
               <alarms:attributeChange name="initiator" newValue="start" oldValue=""/>
            </alarms:attributeChanges>
            </alarms:AlarmAttributeValueChangeInterface>
</alarms:Alarms>
```

Users can have fine grained control over the capability to enable or disable automation for specific scenarios. The configuration is defined in the file `AutomationConfiguration.xml` in the `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf` directory.

Below is the format of the configuration file.

```
<AutomationConfiguration xmlns="http://config.pd.vp.expert.uca.hp.com/">
  <Problems>
    <Problem name="Library_Eltek_Fuse_Fail">
      <GoForAutomation>false</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>false</ManualOverride>
    </Problem>
    <Problem name="Library_Eltek_Battery_Test_Active_2">
      <GoForAutomation>true</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>true</ManualOverride>
    </Problem>
    <Problem name="Library_HW_BaseStationDown">
      <GoForAutomation>true</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>false</ManualOverride>
    </Problem>
    <Problem name="Library_NSN_CoreLinkFailure">
      <GoForAutomation>true</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>false</ManualOverride>
    </Problem>
  </Problems>
</AutomationConfiguration>
```

The Problem names can be a comma separated list in the attribute “name”.

The element <GoForAutomation/> controls the automation at scenario level. It can take the following values.
When set to “true” - automation for the scenario is enabled.
When set to “false” - automation for the scenario is disabled.

When no configuration is defined for a scenario, then by default automation is enabled.

Delayed Automation

Delay for Automation is defined in milliseconds.

```
<DelayForAutomation>1000</DelayForAutomation>
```

After a scenario is eligible for automation (<GoForAutomation/> is true) the delay is applied just before triggering the automation process.

If an alarm waiting for automation to be triggered is not a problem alarm, then automation process is stopped for that alarm.

The automation process is triggered immediately if this configuration is not defined.

Manual Override
The element <ManualOverride/> enables an operator to trigger automation manually.
It can take the following values.

a. true – When set to "true" overrides the global property "manual_mode" in UCAAutomation.properties file.
b. false – When set to "false", the global property "manual_mode" is applied for automation.

When <ManualOverride/> is set to true for a scenario – automation is not triggered until the operator sets the user defined attribute "initiator" to start.

For a scenario if <GoForAutomation/> is set to value "true" and "<ManualOverride/> is also set to value "true", then automation is triggered after operator sets the user defined attribute "initiator" with value "start".

For a scenario if <GoForAutomation/> is set to value "true" and <ManualOverride/> is set to true then Delay for Automation is applied after operator sets the user defined attribute "Initiator" with value "start".

For a scenario if <GoForAutomation/> is set to value "true" and <ManualOverride/> is set to false and "manual_mode" flag in UCAAutomation.properties is set to value "false", then automation is triggered after the delay interval.

4.3 Retry an Action execution

Execution of a failed Action can be retried from the TeMIP Client using a retry code.

HPE SA workflows that are mapped to an action can indicate that the action can be retried by sending back a value of 600 and above in the MinorCode. The failure could be a missing database entry or a wrong configuration or a connectivity error.

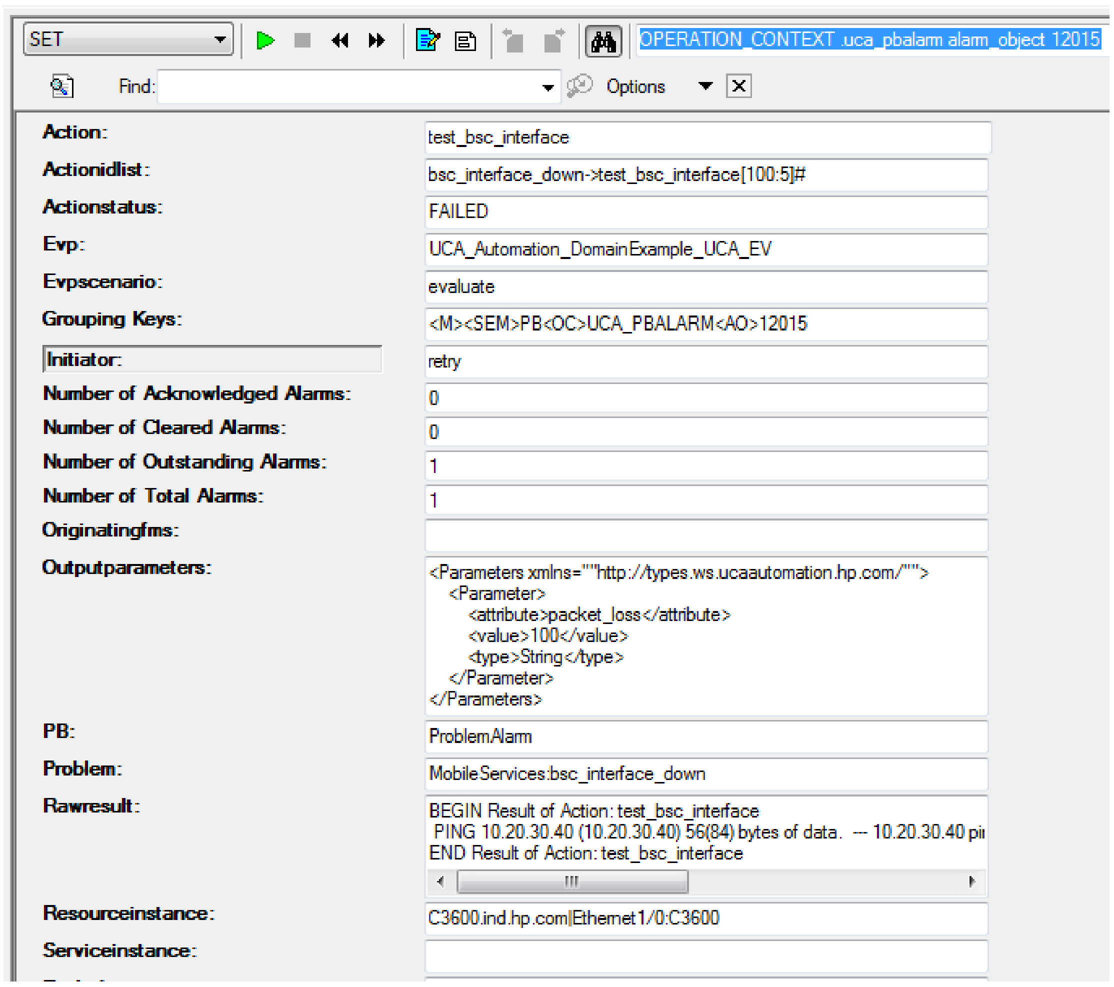
```
<resp_msg xmlns="http://types.ws.ucaautomation.hp.com/">
   <header>
      <ActionId>100</ActionId>
      <TaskId>100</TaskId>
      <ActionInstanceId>1</ActionInstanceId>
      <OrignatorId>operation_context uca_pbalarm alarm_object 4</OrignatorId>
   </header>
```

```
    <body>
      <TaskResponse>
        <MajorCode>
          <Code>501</Code>
          <Description>The test execution failed
          </Description>
        </MajorCode>
        <MinorCode>
          <Code>600</Code>
          <Description>Retry code</Description>
        </MinorCode>
          <TaskStatus>FAILED</TaskStatus>
          <Diagnostics>PING 10.20.30.40 (10.20.30.40) 56(84) bytes of data.
              --- 10.20.30.40 ping statistics
              --- 5 packets transmitted, 0 received, 100% packet loss, time 13999ms
                  </Diagnostics>
         <Parameters>
          <Parameter>
            <attribute>outputparameters</attribute>
            <value>packet_loss,100,String</value>
          </Parameter>
          </Parameters>
      </TaskResponse>
    </body>
</resp_msg>
```

It is entirely up to the workflow designer to have the retry logic in the workflow. By sending back a retry code the operator can take a look at the response and then decide whether the action can be retried. The automation process halts at this step and waits for a user intervention. The retry feature can be disabled by setting the retry_flag as false in the in the `UCAAutomation.properties` file in the `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf` directory.

```
retry_flag=false
```

The Action can be retried by setting the value of the custom attribute Initiator to “retry” from the TeMIP client or by sending an Attribute Value Change message.


SET

OPERATION_CONTEXT .uca_pbalarm alarm_object 12015

Find: Options

Action:	test_bsc_interface
Actionidlist:	bsc_interface_down->test_bsc_interface[100:5]#
Actionstatus:	FAILED
Evp:	UCA_Automation_DomainExample_UCA_EV
Evpscenario:	evaluate
Grouping Keys:	<M><SEM>PB<OC>UCA_PBALARM<AO>12015
Initiator:	retry
Number of Acknowledged Alarms:	0
Number of Cleared Alarms:	0
Number of Outstanding Alarms:	1
Number of Total Alarms:	1
Originatingfms:	
Outputparameters:	
PB:	ProblemAlarm
Problem:	MobileServices:bsc_interface_down
Rawresult:	
Resourceinstance:	C3600.ind.hp.com|Ethernet1/0:C3600
Serviceinstance:	

```
<Parameters xmlns=""http://types.ws.ucaautomation.hp.com/"">
   <Parameter>
      <attribute>packet_loss</attribute>
      <value>100</value>
      <type>String</type>
   </Parameter>
</Parameters>
```

```
BEGIN Result of Action: test_bsc_interface
 PING 10.20.30.40 (10.20.30.40) 56(84) bytes of data.  --- 10.20.30.40 pi
END Result of Action: test_bsc_interface
```

Figure 7: Retry an Action

The custom attribute Actionidlist also gives an indication that an Action is retried

```
bsc_interface_down-
>test_bsc_interface[100:5]#test_bsc_interface[100:5]#test_bsc_interface_failed-
>list_all_available_interfaces[103:6]#
```

After multiple attempts the user has the option of continuing to the next action (in the failed path in the decision tree) or aborting the automation for this scenario. Here is an example of sending an AVC message to continue to next action by using the UCA-EBC alarm injector utility

```
<?xml version="1.0" encoding="UTF-8"?>
<alarms:Alarms xmlns:alarms="http://hp.com/uca/expert/x733Alarm">
        <alarms:AlarmAttributeValueChangeInterface>
        <alarms:identifier>operation_context uca_pbalarm alarm_object 12015</alarms:identifier>
        <alarms:sourceIdentifier>user</alarms:sourceIdentifier>
        <alarms:originatingManagedEntity>osi_system site_site1</alarms:originatingManagedEntity>
        <alarms:additionalText></alarms:additionalText>
        <alarms:attributeChanges>
          <alarms:attributeChange name="initiator" newValue="continue" oldValue=""/>
```

```
            </alarms:attributeChanges>
            </alarms:AlarmAttributeValueChangeInterface>
</alarms:Alarms>
```

In order to abort the automation after multiple retries the Initiator must be set to “abort”

In the AVC message ensure that the source identifier is not set to the values `UCA Automation` or `avcfromevp` as they are internally used

4.4 Protection Switch

In order to protect external systems viz Trouble Ticket or Information management systems accessed by UCA Automation when there is flood and alarms in the zero touch mode (Automatic mode), a concept called Protection Switch has been introduced.

A new UCA-EBC value pack can be written which can monitor incoming Problem Alarms which can potentially access external systems being protected from flood of alarms. A counter of all valid alarms should be maintained and when it reaches a threshold value, the Protection Switch can be turned on in UCA Automation foundation value pack. The switch is represented by a MBean. When switched on, Automation will run in manual mode and all new incoming Problem Alarm needs user intervention to trigger automation process.

Valid alarms are those which have been configured for Automation and in non-manual mode as specified in `AutomationConfiguration.xml` in the `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf` directory.

```
<AutomationConfiguration xmlns="http://config.pd.vp.expert.uca.hp.com/">
  <Problems>
    <Problem name="Library_Eltek_Fuse_Fail">
      <GoForAutomation>false</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>false</ManualOverride>
    </Problem>
    <Problem name="Library_Eltek_Battery_Test_Active_2">
      <GoForAutomation>true</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>true</ManualOverride>
    </Problem>
    <Problem name="Library_HW_BaseStationDown">
      <GoForAutomation>true</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>false</ManualOverride>
    </Problem>
    <Problem name="Library_NSN_CoreLinkFailure">
      <GoForAutomation>true</GoForAutomation>
      <DelayForAutomation>0</DelayForAutomation>
      <ManualOverride>false</ManualOverride>
    </Problem>
  </Problems>
</AutomationConfiguration>
```

In the above example only Problem Alarms “Library_HW_BaseStationDown” and “Library_NSN_CoreLinkFailure” could be considered for incrementing the counter only if `manual_mode=true` in `UCAAutomation.properties` file.

Problem Alarm Library_Eltek_Fuse_Fail is not considered since GoForAutomation is false.

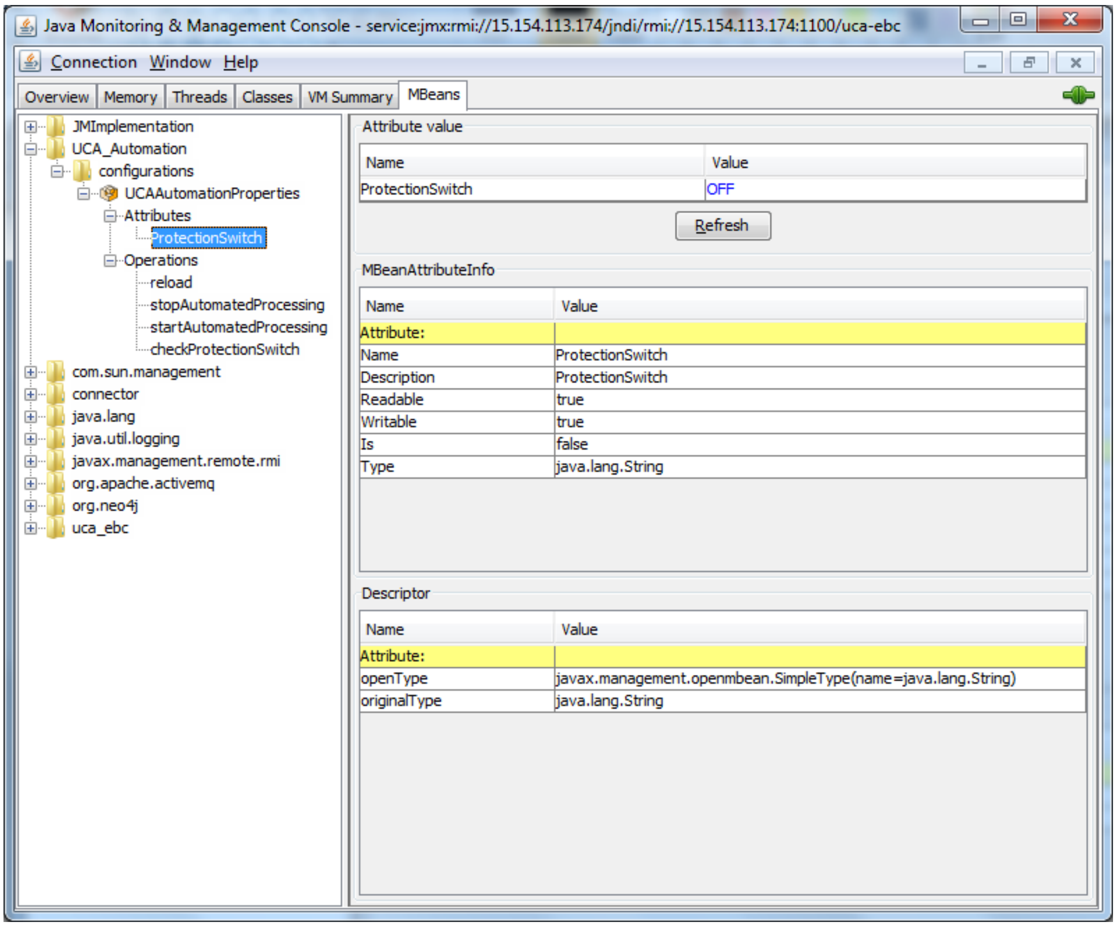
Problem Alarm Library_Eltek_Battery_Test_Active_2 is not considered since ManualOverride is true.

The threshold value and MBean url can be specified in a file.

```
<ExternalConfiguration xmlns="http://config.pd.vp.expert.uca.hp.com/">
  <TTThresholdValue>5</TTThresholdValue>
  <ProtectionSwitchDelay>5000</ProtectionSwitchDelay>
  <AutomationConfigurationFile>/var/opt/UCA-
EBC/instances/default/deploy/UCA_Automation_Foundation_UCA-V2.1-
1A/conf/AutomationConfiguration.xml</AutomationConfigurationFile>
  <AutomationPropertiesFile>/var/opt/UCA-
EBC/instances/default/deploy/UCA_Automation_Foundation_UCA-V2.1-
1A/conf/UCAAutomation.properties</AutomationPropertiesFile>
</ExternalConfiguration>
```

A delay before the Protection value pack turns on the Protection Switch should be added, else it may process incoming Alarms and turn on the switch before the Automation Foundation value pack has a chance to trigger automation process. Then Automation will be turned into manual mode requiring user intervention.

Here is a screen shot of the Protection switch MBean



Attribute value

Name	Value
ProtectionSwitch	OFF

MBeanAttributeInfo

Name	Value
Attribute:	
Name	ProtectionSwitch
Description	ProtectionSwitch
Readable	true
Writable	true
Is	false
Type	java.lang.String

Descriptor

Name	Value
Attribute:	
openType	javax.management.openmbean.SimpleType(name=java.lang.String)
originalType	java.lang.String

Figure 8: Protection Switch MBean

The Protection value pack could use standard jmx API's to turn on the switch. This sets the Protection switch to OFF

```
MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
ObjectName objectName = MBeanRegister.buildObjectname(LOG,
            "UCA_Automation", "configurations", null,
            "UCAAutomationProperties");
mbs.invoke(objectName, "stopAutomatedProcessing", null, null);
```

In the above example ObjectName is constructed using the com.hp.uca.common.jmx.MBeanRegister class found in the uca-common-3.3.jar. This can be used if both the Protection and Automation Foundation value packs reside in the same system.

Below is an example of connecting to Protection MBean in a remote system

```
JMXServiceURL target = new
JMXServiceURL("service:jmx:rmi:///jndi/rmi://15.154.113.174:1100/uca-
ebc");
JMXConnector connector = JMXConnectorFactory.connect(target);
MBeanServerConnection remote = connector.getMBeanServerConnection();
ObjectName mbeanName = new ObjectName("UCA_Automation:type=" +
"configurations"
       + ",name=UCAAutomationProperties");
```

```
remote.invoke(mbeanName, "stopAutomatedProcessing", null, null);
```

The jmx port is configured in using the property uca.ebc.jmx.rmi.port in `${UCA_EBC_INSTANCES}/conf/uca-ebc.properties` file

The criteria to decrement the Problem Alarm counter and ultimately turn off the Protection switch is left to the value pack developer. An admin utility to manually check the status of the switch and turn it off is provided in `${UCA_AUTOMATION_HOME}/Utilities/Admin/bin` (default is /opt/UCA_Automation)

```
[uca@imapptappl016 bin]$ ./uca-auto-admin -ps check
Using UCA for EBC Home directory specified by the environment variable UCA_EBC_HOME:
/opt/UCA-EBC
Using UCA for EBC Data directory specified by the environment variable UCA_EBC_DATA:
/var/opt/UCA-EBC
Enter UCA EBC instance (default is default):
INFO  - Returns the value of the Protection Switch flag
INFO  - Logging to org.slf4j.impl.Log4jLoggerAdapter(org.mortbay.log) via org.mortbay.log.Slf4jLog
INFO  - Protection Switch: ON
[uca@imapptappl016 bin]$

[uca@imapptappl016 bin]$ ./uca-auto-admin -ps OFF
Using UCA for EBC Home directory specified by the environment variable UCA_EBC_HOME:
/opt/UCA-EBC
Using UCA for EBC Data directory specified by the environment variable UCA_EBC_DATA:
/var/opt/UCA-EBC
Enter UCA EBC instance (default is default):
INFO  - Toggles the Protection Switch flag to OFF
INFO  - Logging to org.slf4j.impl.Log4jLoggerAdapter(org.mortbay.log) via org.mortbay.log.Slf4jLog
INFO  - Protection Switch flag is set to OFF
```

4.5 Multiple Resource Instances

Multiple resources can be specified in the custom attribute Resourceinstance of the Problem Alarm. The resources are separated by a '#'

```
(SiteCode,13wc),(Vendor,M2000),(DeviceType,NODEB),(NE_ID,53102):M2000 ne.HW3 NODEB
Whitehills_13WC_53102_2313#(SiteCode,13wf),(Vendor,M2000),(DeviceType,ENODEB),(NE_ID,53107):
M2000 ne.HW3 ENODEB Whitehills_13WC_53107_2313
```

```
C3600.ind.hpe.com|Ethernet1/0:C3600#C4500.ind.hpe.com|Ethernet1/0:C4500
```

4.6 Extensible Automation Action Factory

In UCA Automation external actions (Activation, Alarm and TroubleTicket) actions are performed through an extensible Automation action framework. This is configured in the

`ExternalActionConfig.xml` in the `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf` directory.

UCA Automation comes by default with support for TeMIP directives (Alarm and Trouble Ticket liaison directives) and HPE SA.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActionPolicies xmlns="http://config.pd.vp.expert.uca.hp.com/">
            <!--  NMS Actions -->
            <actions>
               <action name="TeMIP EMS">
                  <actionReference>TeMIP_AO_Directives_localhostNOM</actionReference>
           <actionClass>com.hp.ucaautomation.action.framework.impl.TeMIPAction</actionClass>
               </action>
            </actions>
            <!--  Service Manager Actions -->
            <troubleTicketActions>
               <troubleTicketAction name="TeMIP EMS">
                   <actionReference>TeMIP_TT_Directives_localhostNOM</actionReference>
      <actionClass>com.hp.ucaautomation.action.framework.impl.ServiceManagerAction</actionClass>
                        <properties>
                            <property key="TT_SERVER entity"><value>TT_SERVER .SM</value></property>
                           <property key="CreateTemplateFile"><value>create_tt.xml</value></property>
                           <property
key="AssociateTemplateFile"><value>associate_tt.xml</value></property>
                           <property key="CloseTemplateFile"><value>close_tt.xml</value></property>
                           <property
key="DissociateTemplateFile"><value>dissociate_tt.xml</value></property>
                           <property key="User"><value>temip</value></property>
                           <property key="Input"><value>input</value></property>
                           <!-- property key="Type"><value>SYNCHRONOUS</value></property-->
                           <property key="TT_Template_Prefix_Policy"><value>Service</value></property>
                                    </properties>
                 </troubleTicketAction>
            </troubleTicketActions>

            <!--  Activator Actions -->
            <activationActions>
                <activationAction name="HPSA">
                        <actionReference>HPSA_diagnosticTask_localhost</actionReference>
      <actionClass>com.hp.ucaautomation.action.framework.impl.ServiceActivatorAction</actionClass>
                </activationAction>
            </activationActions>
</ActionPolicies>
```

The TeMIPAction, ServiceManagerAction and ServiceActivatorAction internally uses the Action framework provided by UCA-EBC to execute actions on TeMIP and Service Activator by going through the mediation layer: OSS Open Mediation V7.2. Actions are routed to the mediation using the information stored in the ActionRegistry.xml file.

4.6.1 Action framework Implementation

TeMIPAction

The actions supported by TeMIPAction, which correspond to the Alarm actions supported in Automation Orchestrator Eclipse plugin when modelling Actions

- terminateAlarm
- clearAlarm
- updateAlarm
- checkTroubleTicket
- showAlarm
- setGenericAttribute

ServiceManagerAction

The actions supported by ServicemanagerAction, which correspond to the Trouble Ticket actions supported in Automation Orchestrator Eclipse plugin when modelling Actions

- createTroubleTicket
- closeTroubleTicket
- associateTroubleTicket
- dissociateTroubleTicket

ServiceActivatorAction
The actions supported are. The implementation in this class is used when an Action modelled in Automation Orchestrator Eclipse plugin has dispatch type as HPE SA
- startTask – starts a workflow in HPE SA
- stopTask – stops a running job in HPE SA
- checkTaskStatus – returns the status of a running job in HPE SA.

Example of the TeMIPAction

UCA Automation provides the implementation of the NMSAction class for TeMIP in the AutomationActionsframework.jar (`${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/lib`). Below is an extract of the TeMIPAction class showing how the clearAlarm() method is implemented

```
public class TeMIPAction extends NMSAction{
    @Override
      public Action clearAlarm(Action action, String originatorId,
                  String taskId, String actionId, String
actionExecMode) throws Exception {
            action.addCommand("directiveName", "CLEARALARM");
            //originatorId is the alarm id
            action.addCommand("entityName", originatorId);
            //actionExecMode can be "ASYNCHRONOUS" or "SYNCHRONOUS"
            if (actionExecMode != null &&
actionExecMode.equalsIgnoreCase("ASYNCHRONOUS")) {
                createAndSetCallback(action,
TeMIPActionCallback.class,
                  "clearAlarmCallback", action, taskId, actionId,
originatorId);
            }
            return action;
      }

      public void processClearAlarmResponse(Action action, String
taskId, String actionId, String originatorId) {
            TeMIPActionCallback.processClearAlarmResponse(action,
taskId, actionId, originatorId);
      }
}
```

The method createAndSetCallback is defined and implemented in
com.hp.ucaautomation.action.framework.action.NMSAction
Below is an extract of the TeMIPActionCallback class showing how the clearAlarmCallback method set in the TeMIPAction class, is implemented

```
public class TeMIPActionCallback {
      public static void clearAlarmCallback(Action action,
                  String taskId, String actionId,
                  String originatorId) {
          processClearAlarmResponse(action, taskId, actionId,
originatorId);
      }

      public static void processClearAlarmResponse(Action action,
String taskId,
                  String actionId, String originatorId) {
          String majorCode = "200";
          String majorDescription = null;
          String status = null;
          String diagnostics = null;
          String minorCode = null;
          String minorDescription = null;
          if
(action.getActionStatus().equals(ActionStatus.Completed)) {
              for (ActionResponseItem item :
action.getListActionResponseItem()) {
                  if (item.getRawText() != null) {
                      Element rawtext = item.getRawText().getAny();
            //look for the <ClearAlarm_Success> tag in the action
response
            NodeList nodelist = rawtext
            .getElementsByTagName("ClearAlarm_Success");
            if (nodelist != null && nodelist.getLength() != 0) {
                String clearStatus = nodelist.item(0).getChildNodes()

      .item(0).getNodeValue().trim();
                //200 indicates a success for the status to be
modified in automation console
                majorCode = "200";
                status = "PASSED";
                diagnostics = "CLEARED";
            }
                  }
              }
          } else if
(action.getActionStatus().equals(ActionStatus.Failed)) {
               //501 indicates a failure for the status to be
modified in automation console
              majorCode = "501";
              status = "FAILED";
              diagnostics = "";
              for (ActionResponseItem item :
action.getListActionResponseItem()) {
                  if (item.getRawText() != null) {
                      Element rawtext = item.getRawText().getAny();
            NodeList nodelist = rawtext
      .getElementsByTagName("Generic_Exception");
            if (nodelist != null && nodelist.getLength() != 0) {
                diagnostics =
nodelist.item(0).getChildNodes().item(2)
                  .getChildNodes().item(0).getNodeValue();
            }
                  }
              }
              String actionStatusExplanation =
action.getActionStatusExplanation();
              if(actionStatusExplanation!=null){
                  if(actionStatusExplanation.contains("SOAP Fault")){
                      diagnostics="Failed to Clear
Alarm"+"."+actionStatusExplanation.substring(actionStatusExplanation.
indexOf(":")+1,
actionStatusExplanation.indexOf("Exchange"))+"."+diagnostics;
                      //send back a retry code, since this indicates
a recoverable error
                      minorCode="600";
                      //read the decsription for retry code from a
resource bundle minormessages.properties
                      minorDescription =
ActionCallback.readMinorCodeDescription(minorCode);
                  }else if(actionStatusExplanation.contains("HTTP
transport error")){
                      diagnostics="Failed to Clear
Alarm"+"."+actionStatusExplanation.substring(actionStatusExplanation.
indexOf("HTTP"))+"."+diagnostics;
                      //send back a retry code, since this indicates
a recoverable error
                      minorCode="600";
                      //read the decsription for retry code from a
resource bundle minormessages.properties
                      minorDescription =
ActionCallback.readMinorCodeDescription(minorCode);
                  }
              }
          }
          //read the decsription for major code from a resource
bundle messages.properties
          majorDescription =
ActionCallback.readMajorCodeDescription(majorCode);

          //send response to automation console using a utility
method in the ActionCallback class
          ActionCallback.generateResponseAndSendToConsole(taskId,
actionId,
                        originatorId, majorCode, majorDescription,
status, diagnostics, minorCode, minorDescription);
      }
}
```

4.6.2 Extending the TeMIPAction

If the user wishes to extend the exiting TeMIPAction then its subclass must be specified in the file.

```
<ActionPolicies xmlns="http://config.pd.vp.expert.uca.hp.com/">
  <!--  NMS Actions -->
  <actions>
    <action name="TeMIP EMS">
      <actionReference>TeMIP_AO_Directives_localhostNOM</actionReference>
      <actionClass>com.hp.ucaautomation.action.framework.impl.TeMIPActionExt</actionClass>
    </action>
  </actions>
  <troubleTicketActions>
  …
  </troubleTicketActions>
</ActionPolicies>
```

Here is a sample of how to modify the behavior of updateAlarm(). This method updates the specified "attributeName" with specified "attributeNewValue". In the example the default behavior is modified to update the Alarm field Rawresult by reading an external file.

```
public class TeMIPActionExt extends TeMIPAction {
    public Action updateAlarm(Action action, Alarm alarm, String
originatorId, String attributeName, String attributeNewValue, String
taskId, String actionId, String actionExecMode, String problemName,
            String sourceIdentifier) throws Exception {


   if (alarm != null) {
      // The method is invoked from Automation Foundation Value Pack
internally. use the super
      // class implementation
      super.updateAlarm(action, alarm, originatorId, attributeName,
              attributeNewValue, taskId, actionId, actionExecMode,
              problemName, sourceIdentifier);
   } else {
      // specialized implementation to update rawresult with enriched
information
      //Read an external file and update the Rawresult with this data
      TroubleTicketMapping troubleTicketMapping =
readExternalTTConfig();
      HashMap<String, String> processedData =
preEnrichRawResult(action, originatorId,
             taskId, actionId, sourceIdentifier, problemName,
troubleTicketMapping);
      //this string contains existing data in Rawresult plus enriched
text
      String enrichForTT = (String)processedData.get("enrichForTT");
      //this string contains only the enriched text
      String enrichedText =
(String)processedData.get("enrichedText");
      action.addCommand(“directiveName”, “SET”);
      //originatorId is the alarm id
      action.addCommand(“entityName”, originatorId);
      action.addCommand(“Rawresult”, enrichForTT);

      if (actionExecMode != null
                        &&
actionExecMode.equalsIgnoreCase(“ASYNCHRONOUS”)) {
         //create a callback method to be executed when  //update is
successfull
         createAndSetCallback(action, TeMIPActionCallback.class,
             "updateAlarmCallback", action, taskId, actionId,
      alarmId);
      }
      //if the Rawresult is updated return the enriched text to the
Action framework
      //This will be sent back in the AVC from automation console to
Foundation VP
      LocalVariable localVariable = new LocalVariable();
      localVariable.put("userdata", enrichedText);
      action.setVar(localVariable);
   }
   return action;
}
}
```

Chapter 5
Migrating from earlier versions of UCA Automation

There is no direction migration path from earlier version of UCA automation. The older version of the product has to be uninstalled.

The integration points are

- Between Domain specific correlation value pack and UCA Foundation value pack
- Between HPE SA UCA controller workflow in UCA solution and the workflows in the domain specific solutions

5.1 Migrating from UCA Automation 1.2

After UCA Automation 2.1 is installed, no changes are necessary for integration between Domain specific correlation value pack and UCA Foundation value pack.

Perform the following steps to integrate the UCA HPE SA solution with the domain specific solutions.

1. Select UCA/Parameter -> Workflow Templates view in the HPE SA inventory.
2. Create the mapping to the child domain specific workflows. Create a mapping of a combination of ServiceType and ActionName with the child domain specific workflow, which is designed to handle such scenarios. The HPE SA workflow node used to invoke the domain workflow can be either StartJobAndWait or ExecuteMacro. This must be performed for all the Domain specific actions
3. Three additional parameters log_manager, log_level and sync are passed from UCAController to the domain specific workflows. Define these case-packets in the workflows. The default values of these case-packets are

Table 6: Additional Parameters from UCAController

Case-Packet	Type	Description
log_manager	String	Reference to the Log Manager configured in mwfm.xml. The default value is log_manager
log_level	String	The Log level. The default value is INFORMATIVE
sync	Boolean	Flag indicating that the domain workflow must sync back with UCAController. The default value is true. The destination case-packet in the domain workflow is sync of type Boolean. This case-packet can be used by the domain specific workflow when it syncs back with UCAController.

4. Change the format of the parsed output parameters, if any, sent back by the Domain specific workflows to the UCAController. The new format is as follows

```
<Parameters><Parameter><attribute>attribute1</attribute><value>value1
</value><type>attribute1
```

```
type</type></Parameter><Parameter><attribute>attribute2</attribute><v
alue>value2</value><type>attribute2
type</type></Parameter>...</Parameters>
```

```
Eg
```

```
<Parameters>
<Parameter>
    <attribute>packet_loss</attribute>
    <value>100</value>
  <type>String</type>
</Parameter>
</Parameters>
```

5. Follow the steps specified in the next section for changes necessary in UCA Automation foundation value

5.2 Migration from UCA Automation 2.0

1. Install all the latest patch for UCA-EBC 3.3 (Server patch 00002 is mandatory)
2. Install UCA Automation 2.1
3. Backup the UCA Automation foundation value pack 2.0 and undeploy it. Deploy the Automation foundation value pack 2.1 and configure it as specified in the Installation guide. Add all the scenario specific configuration (from the backup) if any into the AutomationConfiguration.xml in `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf` folder. Add all the mappings of multiple Problems to a single generic Problem (from the backup) if any into the mappers.xml in `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/requestresponse` folder.
4. Ensure that the OrchestraConfiguration.xml and uca-ebc.properties file in `${UCA_EBC_INSTANCES}/conf/` is modified to specify the correct version of UCA Automation Foundation Value Pack
5. Edit the `config.properties` file in the `${NOM_INSTANCE}/ips/uca-autoconsole-ca-20/etc` directory and specify the correct version of UCA Automation Foundation Value Pack in the property `uca.console.service`
6. Retain only the profile named “ucaatm” in the `${UCA_EBC_INSTANCES}/conf/GraphDisplayProfiles.xml.` Update the “ucaatm” profile with the one provided in `<UCA Automation root>/Utilities/TomSawyerVisualization`
7. Restart UCA-EBC.
8. Please follow the install guide if you switch to UMB.
9. Run the following SQL statement to add a new state transition in UCA_STATEMAPPING table. Connect to the database configured in `${UCA_EBC_INSTANCES}/deploy/UCA_Automation_Foundation_UCA-V2.1-1A/conf/UCAAutomation.properties`
   Insert into UCA_STATEMAPPING (CURRENTSTATE,SARESPONSE,NEXTSTATE,ISPARENT__) values ('Request_Sent','500','Failure','0');

Chapter 6
UCA Automation demo scenario

The UCA Automation kit contains a demo of the automation in the following scenario:

- A series of cell down alarms are generated.
- This storm of alarms is interpreted and the problem is isolated as the BSC is down.
- UCA Automation performs a test to verify whether the BSC is not working or is a false positive.
- If the BSC is down, the system performs a test to check all the available free interfaces.
- Later, the system triggers an action to recover the service, switching it to an available interface.
- After a successful recovery, the alarm is updated with recovery information and all open trouble tickets are closed.
- After a failure from recovery, the alarm is updated with diagnostic information and a trouble ticket is opened.

6.1 Performing the demo scenario

Follow the procedure to run the UCA Automation demo scenario.

1. Deploy the HPE SA demo value pack `UCA_HPSA_DomainExample_VP-V21-1A.zip` available under `/opt/UCA_Automation/UCA_Automation_HPSA_VPs` after installation.
2. Deploy the UCA demo value packs `UCA_Automation_DomainExample_UCA_PD-vp-V2.1-1A.zip` and demo evaluate value pack `UCA_Automation_DomainExample_UCA_EV-vp-V2.1-1A.zip`.

   These value packs are available at `/opt/UCA_Automation/UCA_Automation_UCA_VPs`.
3. Database configuration file for the UCA demo value pack.

   Modify the `DBConfiguration.xml` in `/var/opt/UCA-EBC/instances/default/deploy/UCA_Automation_DomainExample_UCA_EV-V2.1-1A/conf`.

   The contents of the file are as follows. Specify the database name, URL, username, and password.

```
#contains the Inventory database access parameters
<DBConfiguration>
   <database>postgres</database>
 <postgressUrl>jdbc:postgresql://localhost:5444/postgres
 </postgressUrl>
   <oracleUrl>
jdbc:oracle:thin:@localhost:1521:hpsadb
 </oracleUrl>
   <username>hpsa61</username>
    <password>hpsa61</password>
</DBConfiguration>
```

4. Configure the action reference in UCAEvaluate.properties as follows

```
#UMB
#actionReference=TeMIP_AO_Directives_localhost
#NOM
```

```
actionReference=TeMIP_AO_Directives_localhostNOM
```

5. Upload the example Decision Tree `/opt/UCA_Automation/Utilities/DecisionTree/etc/DomainEx/DecisionTree.xml` using the command line Decision Tree utility
6. Edit the `${UCA_EBC_INSTANCES}/conf/OrchestraConfiguration.xml` file and add the following route configuration

```
<Routes>
        <Route name="Copy from UCA Automation Foundation VP to from UCA Automation EV VP
">
         <COPY>
          <Source>
        <ValuePackNameVersion>UCA_Automation_Foundation_UCA-V2.1-
1A</ValuePackNameVersion>

        <ScenarioName>UCA_Automation_Foundation_UCA.requestresponse</ScenarioName>
        </Source>
        <Destinations>
          <Destination>
                <Target>
                <ValuePackNameVersion>UCA_Automation_DomainExample_UCA_EV-V2.1-
1A</ValuePackNameVersion>

<ScenarioName>UCA_Automation_DomainExample_UCA_EV.evaluate</ScenarioName>
        </Target>                                                </Destination>
        </Destinations>
        </COPY>
        </Route>
        <Route name="Copy from UCA Automation EVP to from UCA Automation Foundation VP ">
          <COPY>
            <Source>
          <ValuePackNameVersion>UCA_Automation_DomainExample_UCA_EV-V2.1-
1A</ValuePackNameVersion>

        <ScenarioName>UCA_Automation_DomainExample_UCA_EV.evaluate</ScenarioName>
                </Source>
                <Destinations>
                <Destination>
                  <Target>
        <ValuePackNameVersion>UCA_Automation_Foundation_UCA-V2.1-
1A</ValuePackNameVersion>
        <ScenarioName>UCA_Automation_Foundation_UCA.requestresponse</ScenarioName>
                  </Target>
                </Destination>
                </Destinations>
          </COPY>
        </Route>
 </Routes>
```

7. Edit the `${UCA_EBC_INSTANCES}/conf/uca-ebc.properties` file and allow loops in the cascading workflow. Restart UCA-EBC

```
uca.ebc.orchestra.loops.allowed=true
```

8. Generate alarms from TeMIP, which match the pattern present inside the PD value pack `UCA_Automation_DomainExample_UCA_PD-vp-V2.1-1A.zip`. Ensure that the TeMIP operation contexts `uca_network` and `uca_pbalarm` are created. Refer the UCA Automation Installation guide for more details. Run the sample alarm generation utility script provided with the Domain example PD value pack in the bin folder.

9. Interact with the Tasks from the UCA Automation console.