hp Unified Correlation Analyzer

Unified Correlation Analyzer
for EBC
Inference Machine

Version 3.2

Release Notes

Edition: 1.0

For Windows© and Linux (RHEL 5.9 & 6.5) Operating Systems

April 2015

Legal Notices

Warranty

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

License Requirement and U.S. Government Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Adobe®, Acrobat® and PostScript® are trademarks of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Java™ is a trademark of Oracle and/or its affiliates.

Microsoft®, Internet Explorer, Windows®, Windows Server®, and Windows NT® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

Red Hat® is a registered trademark of the Red Hat Company.

Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

Contents

Tables

Preface

These Release Notes describe critical information related to the HP UCA for EBC Problem Detection product.

Product Name: Unified Correlation Analyzer for EBC Problem Detection

Product Version: 3.2

Kit Version: V3.2

Please read this document before installing or using this Software.

Intended Audience

Here are some recommendations based on possible reader profiles:

- Solution Developers
- 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:

Product Version	Supported Operating systems
UCA for Event Based Correlation Development Kit Problem Detection Extension Version 3.2	• Windows XP / Vista • Windows Server 2007 • Windows 7 • Linux Red Hat Enterprise Linux Server release 5.9 & 6.5
UCA for Event Based Correlation Development Kit Topology State Propagator Extension Version 3.2	• Windows XP / Vista • Windows Server 2007 • Windows 7 • Linux Red Hat Enterprise Linux Server release 5.9 & 6.5

Table 1 - Software versions

Typographical Conventions

`Courier` Font:

- Source code and examples of file contents.
- Commands that you enter on the screen.
- Pathnames
- Keyboard key names

Italic Text:

- Filenames, programs and parameters.
- The names of other documents referenced in this manual.

Bold Text:

- To introduce new terms and to emphasize important words.

Associated Documents

- HP UCA for EBC IM – Installation Guide
- HP UCA for EBC IM – User Guide
- HP UCA for EBC PBD – TeMIP Client Guide
- HP UCA for EBC - Installation Guide
- HP UCA for EBC - Administration, Configuration, and Troubleshooting Guide
- HP UCA for EBC - Reference Guide
- HP UCA for EBC - Topology Extension Guide
- HP UCA for EBC - Value Pack Development Guide
- HP UCA for EBC - User Interface Guide

Support

Please visit our HP Software Support Online Web site at https://softwaresupport.hp.com/ for contact information, and details about HP 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

Main changes since last release

Previous officially released version of this product was UCA for EBC Problem Detection V3.1

UCA for EBC Problem Detection is now part, along with UCA for EBC Topology State Propagator, of UCA for EBC Inference Machine.

Since delivery of UCA for EBC Problem Detection V3.1 the following features and fixes have been implemented.

1.1 Java versions

The Problem Detection kit is no more built on the JDK 1.6. Hence, the target binary of the kit does not support JRE 1.6 anymore.

The only supported Java platform is JDK/JRE 1.7.

1.2 IM SDK Eclipse wizard is now available

The IM SDK Eclipse wizard is now available offering templates for:

- UCA EBC VP with or without topology
- PBD VP with or without topology template
- TSP VP (topology mandatory)
- IM VP (PBD scenario + TSP scenario) (topology mandatory)

1.3 Trouble Ticket feature implementation for TSP service alarms

1.4 Problems initialization has changed

Problems initialization is done in the following way:

- ProblemDefault values from configuration apply to all other Problems now, only if they were overwritten in the configuration.
- Certain fields, like Strings, Booleans, and Longs defined in the default problem are now valid for all the other Problems.

The same mechanism applies also for Propagations initialization in TSP.

1.5 Topology Generic Queries implementation

The SDK brings of UCA-EBC brings generic queries in package `com.hp.uca.expert.topology.query` which holds the classes:

- GenericQuery
- NodeQuery
- NodeWithCountQuery

- RelationQuery

Particularly the class `GenericQuery` is intensively used in IM framework as it can return multiple values from a single Cypher query.

1.6 SDK

The Problem Detection SDK does not exist anymore. It has been replaced by the Inference Machine SDK.

1.6.1 Refactoring of some classes/packages

A new library uca-evp-im-common.jar has been introduced. It holds common packages for developers using not only Problem Detection kit, but also the Topology State Propagator kit.

1.6.2 Package

The package UCA-EBC-DEVPD does not exist anymore. It has been merged into a bigger package named UCA-EBC-DEVIM, which contains the whole Inference Machine product elements, i.e.:

- The consolidated javadoc for Problem Detection (PD), Topology State Propagator (TSP) and uca-evp-im-common library.
- The skeletons for Problem Detection Value Pack, Topology State Propagator Value Pack and Inference Machine (PD and TSP scenarios) Value Pack
- The schemas used in Inference Machine (1 for PD, 1 for TSP, 1 for common)
- Various examples: pd-example and im-example.

1.6.3 Generic Events (other than Alarm types) are supported

Problem Detection is now able to correlate generic events and group them. Hence:

- The Trigger of a PD correlation group can be now an Event (type introduced in UCA-EBC V3.1)
- Most methods are applicable therefore for the Event type as parameter and not only Alarm. That explains why some methods are now deprecated.

1.6.4 New ways for computing Problem Information

When new alarm comes in Problem Detection, Problem information is now computed in 2 ways:

1.6.4.1 Case where Problem Detection is topology-aware

In such a case, the following conditions are checked by default:

- *MainPolicy.enableTopoAccess* attribute is set to true
- the *CypherQuery* tag is present in the passing filter tags parameters and should provide the name of the Cypher Query to execute

If conditions are passed, both methods **GeneralBehaviourDefault.computeSourceUniqueId (Event event)** and **ProblemDefault.computeDbRecords(String dbUniqueIdReference, Event event)** are used to compute the Problem Alarm information.

Notes:

- The above default conditions can be changed by overriding the *ProblemDefault.isAllowingDbAccess(Event event)* method.
- In case of successful computation, method *ProblemDefault.computeProblemEntity(Event event)* is therefore not used.

1.6.4.2 Default case (non-topology aware)

If above case does not apply or fails, the new *ProblemDefault.computeProblemEntity(Event event)* is used.

1.6.5 ProblemXmlConfig schema changes

The *ProblemXmlConfig.xml* configuration file was modified concerning the following elements:

1.6.5.1 Namespace

Some elements defined in the *ProblemXmlConfig.xml* configuration file are now coming from a common schema with a different namespace. Hence, existing configuration file should be migrated. Refer to “2.1 How do I migrate my PD VP 3.0/3.1 to 3.2?” migration steps for more information.

1.6.5.2 MainPolicy

- New attribute *enablePrioritySort* : Boolean flag indicating whether the groups should be sorted on priority order or not. Default is false,
- New attribute *multipleParentSupport* : Boolean flag indicating whether an alarm grouping will send the parent relationship only for the highest priority parent (false), or for each of the ProblemAlarm where this alarm is grouped (true). Default is true,
- New attribute enableTopoAccess : Boolean flag indicating whether to use topologyAccess when computing information for Problem Alarm (by calling computeSourceUniqueID(Event event) and computeDBRecords() methods) during the workflow) (true) or not (false). Default is false. When true, the computeProblemEntity(Event event) is not called. Attention, this uses Neo4j database, so requires Topology license.

1.6.5.3 ProblemPolicy

- New attribute *enableComputeProblemEntityFromMappers*: When true, enables the use of calling mappers in computeProblemEntity(). Default is true,
- New attribute *enableComputeProblemEntityFromFields*: When true, enables calculation of fields key/value pairs in computeProblemEntity(). Default is false,
- New element *computeProblemEntityFromFields*: Configuration of the FieldsChooser element, which is a sequence of fields to use as keys. Used in computeProblemEntity() when calculation of fields key/value pairs is enabled and when ComputeProblemEntityFields tag is not used.

1.6.6 ProblemDefault.computeProblemEntity(Event event)

This is a new method that takes Event as parameter. It is called by the existing *computeProblemEntity(Alarm alarm)* method. The default behavior of the new *computeProblemEntity(Event)* method has been completely improved to satisfy most of the end-user needs. It executes the following procedures (1.6.6.1, 1.6.6.2 and 1.6.6.3) in respective order.

1.6.6.1 Usage of extended mappers

Firstly, it makes use of the new UCA-EBC V3.2 feature: the extended mappers.

When an event comes in the ProblemDetection value pack, it is checked against the presence of the filter tag named **"ComputeProblemEntityMappers"** which is a parameter tag that should contain the name of the mapper(s) to use for computing the problem entity.

If the tag is present in the incoming filtered alarm, and if the mappers referenced in this tag are well defined, the mappers are executed against the incoming alarm and the result of each mapper is used as one element of the problem entity list returned by this function.

The usage of extended mappers is automatically taken into account.

Notes about mappers' usage:

- The mappers usage can be disabled by setting the corresponding **ProblemPolicy.enableComputeProblemEntityFromMappers** attribute to **false** in *ProblemXmlConfig.xml* file. By default, it is considered as *true*.
- Each mapper name in the *"ComputeProblemEntityMappers"* tag should be separated by **"."**.
- You can change the name of the filter tag used by overriding the *getProblemEntityMappersTag()* method of your problem.

1.6.6.2 Direct mapping of alarm fields as key/value pairs

Secondly, if requested, it can make use of the fields of the alarm computed as key/value pairs. This function work as described below, each option being evaluated in following order:

1. Use of a well-known tag

If the filter tag **"ComputeProblemEntityFields"** is present in the incoming alarm filtered tags, that tag should contain the name of the field(s) to use for computing the problem entity. Each field described in this tag is checked against its presence in the alarm and the resulted problemEntity is computed as **$field.name$separator$field.value**.

Notes about ComputeProblemEntityFields filter tag usage:

- The computation of the key/value pairs can be enabled by setting the corresponding **ProblemPolicy.enableComputeProblemEntityFromFields** attribute to **true** in *ProblemXmlConfig.xml* file. By default, it is considered as *false*, hence this feature is by default not used.
- Each field name in the *"ComputeProblemEntityFields"* tag should be separated by **"."**.
- You can change the name of the filter tag used by overriding the *getProblemEntityFieldsTag()* method of your problem.
- You can change the value of **$separator** used by overriding the *getProblemEntitySeparator()* method of your problem. By default, it is **"="**.

2. Use of new policy

The corresponding **ProblemPolicy.computeProblemEntityFromFields** element can be defined in *ProblemXmlConfig.xml* file and is used for computing the problem entity. This policy defines a sequence of XML fields elements and a **keyValueSeparator** XML element which is by default **"="**.

Each field described in this XML element is used as one element of the problem entity list returned by the *computeProblemEntity()* method. Each field is defines either a **tagName**, either a **fieldName**.

- When **tagName** is defined, it corresponds to a tag that should be present if the incoming alarm filtered tags which should define the field of the alarm to take into account.

It is then checked against its presence in the alarm filtered tags and the resulted problemEntity is computed as **$alarmField$keyValueSeparator$alarmField.value**, where **$alarmField** should be present in the alarm and is equivalent to **$field.key.tagName.value**

- When **tagName** is not defined and **fieldName** is defined, it corresponds directly to the field of the alarm to take into account.

  The field name is then checked against its presence in the alarm and the resulted problemEntity is computed as **$fieldName$keyValueSeparator$fieldName.value**

Notes about computeProblemEntityFromFields policy usage:

- The computation of the key/value pairs can be enabled by setting the corresponding **ProblemPolicy.enableComputeProblemEntityFromFields** attribute to **true** in *ProblemXmlConfig.xml* file. By default, it is considered as *false*, hence this feature is by default not used.
- If the filter tag **"ComputeProblemEntityFields"** is present in the incoming alarm filtered tags, it supersedes the policy, hence the policy is not used.
- You can ignore a specific value for each field using the **valueIgnored** XML element associated to it.

1.6.6.3 Default mode

When none of above two methods is used, the function returns as previously (up to V3.1) the originating managed entity of the incoming Alarm.

1.6.6.4 Modification of examples

The classes *Problem_Synch* and *Problem_BitError* are now showing the usage of extended mappers feature to compute their problem entity based on bsc and bts identifiers. The *computeProblemEntity()* function has then been removed from those classes, which are now using the mapper **getBscBtsFromUserText** instead.

1.6.7 GeneralBehaviourDefault.computeSourceUniqueId (Event event)

This method is used to calculate the unique identifier from information source stored in the event. It is called when Problem Detection is topology-aware, i.e. when the MainPolicy.enableTopoAccess attribute is set to true. In such a case, a special filter should be defined with the ReservedForGeneralBehavior as the filter name. Inside this filter, the ComputeSourceUniqueIdMapper tags are used to compute the source unique Id. When mappers are defined in the topFilter having the name ReservedForGeneralBehavior, Problem Detection will call the computeSourceUniqueId(Event) method.

Example (extracts of filters and mappers files):

```
 <topFilter name="ReservedForGeneralBehavior">
    <anyCondition>
        <anyCondition tag="PATTERN_Mappers">
            <allCondition tag="ComputeSourceUniqueIdMapper=NodeB_UniqueID_1">
                <instanceOfFilterStatement>

<fullClassName>com.hp.uca.expert.alarm.AlarmCommon</fullClassName>
                </instanceOfFilterStatement>
                <stringFilterStatement>
                    <fieldName>additionalText</fieldName>
                    <operator>contains</operator>
                    <fieldValue>PowerAntenna</fieldValue>
                </stringFilterStatement>
```

```
            </allCondition>
            <allCondition tag="ComputeSourceUniqueIdMapper=NodeB_UniqueID_2">
                <instanceOfFilterStatement>

<fullClassName>com.hp.uca.expert.alarm.AlarmCommon</fullClassName>
                </instanceOfFilterStatement>
                <stringFilterStatement>
                    <fieldName>additionalText</fieldName>
                    <operator>contains</operator>
                    <fieldValue>DIP_Failure</fieldValue>
                </stringFilterStatement>
            </allCondition>
        </anyCondition>
    </anyCondition>
 </topFilter>

 <mapper name='NodeB_UniqueID_1'>
        <pattern>
       <expression>[btsID]~[location]</expression>
     <matcher>(.*)</matcher>
     <mappedTo>$1</mappedTo>
   </pattern>
 </mapper>
```

1.6.8 ProblemDefault.computeDbRecords(String dbUniqueIdReference, Event event)

This method is used to calculate the Neo4j query, which will be executed to retrieve the data base records for having the database id reference for the Event. Called by the Problem Detection Framework when the *MainPolicy.enableTopoAccess* attribute is set to true and when *CypherQuery* tag is present.

Example (extracts of filters and mappers files):

```
    <anyCondition tag="ProblemAlarm,CypherQuery=GetCellFromNodeBOrBts">
        <allCondition>
            <instanceOfFilterStatement>
                <fullClassName>com.hp.uca.expert.alarm.AlarmCommon</fullClassName>
            </instanceOfFilterStatement>
            <stringFilterStatement>
                <fieldName>userText</fieldName>
                <operator>matches</operator>
                <fieldValue><![CDATA[.*<action>UCA EBC
.*</action><trigger>.*</trigger><group>.*</group>.*]]></fieldValue>
            </stringFilterStatement>
            <stringFilterStatement>
                <fieldName>additionalText</fieldName>
                <operator>contains</operator>
                <fieldValue>PowerAntenna</fieldValue>
            </stringFilterStatement>
        </allCondition>
    </anyCondition>

  <cypherQuery name='GetCellFromNodeBOrBts'>
    <query><![CDATA[START startNode=node:NodeBsByUniqueId(uniqueId =
{nodeUniqueId})
        MATCH (startNode)-[relation:ServingCell]->(endNode)<-[?:ServingCell]-
(endNodeRelatives)
```

```
        RETURN startNode, relation,endNode, endNode.domain, endNode.type,
endNode.uniqueId, count(endNodeRelatives)]]>
    </query>
  </cypherQuery>
```

1.6.9 ProblemDefault.computeGroupPriority(Event event)

A default implementation has been introduced to make use of specific tags that can be set at filters level: "**Bundle.Priority**" which defines the priority of the family of Problems and "**Problem.Priority**" which defines the priority of the Problem. The values for these tags should be numeric.

If one of those tags are present after filtering an alarm, the group priority is computed using the formula:

*Bundle.Priority * $priority.factor + Problem.Priority*

If none of the tags is present, the group priority is left to *null*.

The group priority is automatically taken into account if the attribute **enablePrioritySort** is defined to **true** in MainPolicy of ProblemXmlConfig.xml file. It means that all calls to scenario.getGroups().getAllGroups() or to scenario.getGroups().getGroupsWhereXXX() will return the groups sorted on priority.

By default, the attribute **enablePrioritySort** is considered as **false** if not present, hence groups are not sorted by default.

Notes about the priority computation:

Lower priority numbers come first. A *null* priority comes last.

You can change the value of the *$priority.factor* used by overriding the *getBundlePriorityFactor()* method of your problem.

You can change the name of the *Bundle.Priority* tag used by overriding the *getBundlePriorityTag()* method of your problem.

You can change the name of the *Problem.Priority* tag used by overriding the *getProblemPriorityTag()* method of your problem.

1.6.9.1 Example with Alarm

Trigger alarm A1 comes in with *Bundle.Priority=10,Problem.Priority=1* => group G1 priority will be set to 10001.
Trigger alarm A2 comes in with *Problem.Priority=2* => group G2 priority will be set to 2.
Trigger alarm A3 comes in with no tags => group G3 priority will be set to *null*.
Now suppose an alarm S is subalarm of all 3 above groups => the getGroups().getGroupsWhereAlarmSetAs(S, Qualifier.SubAlarm) will return the groups [G2, G1, G3] in strict order if *MainPolicy.enablePrioritySort* is set.

1.6.9.2 Example with Event (other than Alarm)

Trigger event E1 comes in with *Bundle.Priority=10,Problem.Priority=1* => group G1 priority will be set to 10001.
Trigger event E2 comes in with *Problem.Priority=2* => group G2 priority will be set to 2.
Trigger event E3 comes in with no tags => group G3 priority will be set to *null*.
Now suppose an event S is subEvent of all 3 above groups => the getGroups().getGroupsWhereEventSetAs(S, EventQualifier.SubEvent) will return the groups [G2, G1, G3] in strict order if *MainPolicy.enablePrioritySort* is set.

1.6.10 ProblemDefault.computeTimeWindow(Event event)

The default behavior of the default computeTimeWindow(Alarm alarm) method has been changed to make use of specific tag "Trigger.TimeLimit.Seconds" that can be set at filters level and can be applied on the Event generic type.

If this tag is present after filtering an alarm, and given that the value is T, the timeWindow returned overrides the one defined at ProblemPolicy level and is computed as:

If T is 0 : TimeWindowMode.NONE
If T is not 0 : TimeWindowMode.TRIGGER and Window is [abs(T) * 1000 , abs(T) * 1000]
Note: you can change the name of the Trigger.TimeLimit.Seconds tag used by overriding the getTriggerTimeLimitSecondsTag() method of your problem.

1.6.11 Deprecated APIs

All methods/classes/packages below are deprecated with this version and will be removed in next major update.

This is mainly due to the fact that most of the methods are now coming within uca-evp-common.jar that is used also by another toolkit (aka Topology State Propagator for Service Impact).

Type	API	Deprecated by
Package	com.hp.uca.expert.vp.pd.core.exception	com.hp.uca.expert.vp.common.exceptions
Method	ProblemDefault.computeDelayForTroubleTicketCreation(Alarm alarm)	ProblemDefault.computeDelayForTroubleTicketCreation(Event event)
Method	ProblemDefault.computeDelayForProblemAlarmCreation(Alarm alarm)	ProblemDefault.computeDelayForProblemAlarmCreation(Event event)
Method	ProblemDefault.computeDelayForProblemAlarmClearance(Alarm alarm)	ProblemDefault.computeDelayForProblemAlarmClearance(Event event)
Method	ProblemDefault.computeTimeWindow(Alarm alarm)	ProblemDefault.computeTimeWindow(Event event)
Method	PD_Service_Enrichment.setAlarmIsMissingInformation(Alarm a, String problemName)	PD_Service_Enrichment.setEventIsMissingInformation(Event e, String problemName)
Method	PD_Service_Enrichment.setAlarmIsNoMoreMissingInformation(Alarm a, String problemName)	PD_Service_Enrichment.setEventIsNoMoreMissingInformation(Event e, String problemName)
Method	PD_Service_Enrichment.isAlarmMissingInformation(Alarm a, String problemName)	PD_Service_Enrichment.isEventMissingInformation(Event e, String problemName)
Method	PD_Service_Enrichment.requestAlarmComputation(Scenario scenario, Alarm a)	PD_Service_Enrichment.requestEventComputation(Scenario scenario, Event e)
Method	PD_Service_Group.calculateLeadGroup(CollectionGroup groups)	PD_Service_Group.calculateLeadGroup(Collection<Group> groups, boolean sorted)
Method	PD_Service_Group.isLeadGroup(Group potentialLeaderGroup, CollectionGroup groups)	PD_Service_Group.isLeadGroup(Group potentialLeaderGroup, Collection<Group> groups, boolean sorted)
Method	PD_Service_Lifecycle.cloneAlarmToBeReEvaluated(Alarm alarm)	PD_Service_Lifecycle.cloneEventToBeReEvaluated(Event event)

Type	API	Deprecated by
Method	PD_Service_Util.extractSubString()	com.hp.uca.expert.vp.common.services.UtilService.extractSubString()
Method	PD_Service_Util.retrieveBeanFromContextXml()	com.hp.uca.expert.vp.common.services.UtilService.retrieveBeanFromContextXml()
Method	PD_Service_Util.fileFromResourceName()	com.hp.uca.expert.vp.common.services.UtilService.fileFromResourceName()
Method	PD_Service_Util.storeProblemInfosInAlarmLocalVariable(ProblemContext problemContext, Alarm alarm, ListProblemInfo problemInfos)	PD_Service_Util.storeProblemInfosInEventLocalVariable(ProblemContext problemContext, Event event, List<ProblemInfo> problemInfos)
Method	PD_Service_Util.retrieveProblemInfosFromAlarmLocalVariable(ProblemContext problemContext, Alarm alarm)	PD_Service_Util.retrieveProblemInfosFromEventLocalVariable(ProblemContext problemContext, Event event)
Class	TestUtils	com.hp.uca.expert.vp.common.testmaterial.TestUtils

1.7 StateListener added for asserting States in Junits

A StateListener has been added to the `com.hp.uca.expert.testmaterial` package. This class can be used to assert actions done on States, as AlarmListener is used for asserting actions on Alarms. The StateListener is to be used in Junits of an Inference Machine Value Pack, for the TSP scenario's States. The Junit AbstractJunitIntegrationTest of the `com.hp.uca.expert.testmaterial`, which is the tool box that helps the development of Junit Tests for UCA-EBC, has been enhanced with a StateListener. Therefore, this class provides now the following extra methods:

- `waitingForStateInsertion(StateListener stateListener, long period, long maxTimeBeforeTimeout)`
- `waitingForStateRetract(StateListener stateListener, long period, long maxTimeBeforeTimeout)`
- `waitingForStateUpdate(StateListener stateListener, long period, long maxTimeBeforeTimeout)`

1.8 Inference Machine custom Lifecycle classes

The class *com.hp.uca.expert.vp.common.lifecycle.MixEventsAndStateLifeCycleExtended.class* has been added in the ***uca-evp-im-common.jar*** common library. This class is an enriched Alarm Lifecycle class, managing both States and others Events (Alarms and other events) lifecycle. Alarms passing just the top filter "*ReservedForGeneralBehavior*" will not be inserted in the Working Memory.
For the Topology State Propagator scenario, as well as for the PD scenario, in the IM Value Pack, there are two new classes extending this common class:

- The *com.hp.uca.expert.vp.pd.im.lifecycle.InferenceMachineLifeCycleExtended* is used as the Problem Detection scenario extended life cycle in an Inference Machine valuepack. This class handles alarms, events and states lifecycle and it will bypass service alarms received from the network.

- The *com.hp.uca.expert.vp.tp.im.lifecycle.InferenceMachineLifeCycleExtended* is used as the Topology State Propagator scenario extended life cycle in an Inference Machine valuepack. This class handles alarms, events and states lifecycle.

Chapter 2

Migration steps from V3.1 to V3.2

PD 3.2 is now part of the Inference Machine, which embeds PD and TSP products. As PB and TSP have the exact same needs to execute actions on NMS (create alarm, clear alarm, group alarms, etc.), it has been decided to use a common ActionsFactory for this.

This common ActionsFactory is now part of a common library, which is delivering its own namespace.

As this namespace is different, the compatibility is broken but in counterpart, it brings some improvements:

- the logic of actions is separated from PD and TSP
- as such, it is reusable easily (same ActionsFactory can be used across PD and TSP)
- easier to understand at the end

2.1 How do I migrate my PD VP 3.0/3.1 to 3.2?

Problem Detection 3.2 **does not provide any automatic migration tool** for your Java files.

However, the SDK provides an XLST (eXtensible Stylesheet Language Transformation) file that you can use to migrate your PD configuration file.

2.1.1 In your Java code

2.1.1.1 Removed classes

Following imports will generate compilation errors because the classes do not exist anymore

Class (in V3.1)	Should be replaced in V3.2 by
`import com.hp.uca.expert.vp.pd.config.Action`	`import com.hp.uca.expert.vp.im.config.Action`
`import com.hp.uca.expert.vp.pd.config.Actions`	`import com.hp.uca.expert.vp.im.config.Actions`
`import com.hp.uca.expert.vp.pd.config.BooleanItem`	`import com.hp.uca.expert.vp.im.config.BooleanItem`
`import com.hp.uca.expert.vp.pd.config.Booleans`	`import com.hp.uca.expert.vp.im.config.Booleans`
`import com.hp.uca.expert.vp.pd.config.LongItem`	`import com.hp.uca.expert.vp.im.config.LongItem`
`import com.hp.uca.expert.vp.pd.config.Longs;`	`import com.hp.uca.expert.vp.im.config.Longs`
`import com.hp.uca.expert.vp.pd.config.StringItem;`	`import com.hp.uca.expert.vp.im.config.StringItem`
`import com.hp.uca.expert.vp.pd.config.Strings`	`import com.hp.uca.expert.vp.im.config.Strings`

Class (in V3.1)	Should be replaced in V3.2 by
import com.hp.uca.expert.vp.pd.config.TroubleTicketAction	import com.hp.uca.expert.vp.im.config.TroubleTicketAction
import com.hp.uca.expert.vp.pd.config.TroubleTicketActions	import com.hp.uca.expert.vp.im.config.TroubleTicketActions
import com.hp.uca.expert.vp.pd.core.exception.InvalidSupportedActions	import com.hp.uca.expert.vp.common.exceptions.InvalidSupportedActions
import com.hp.uca.expert.vp.pd.core.exception.InvalidSupportedTroubleTicketActions	import com.hp.uca.expert.vp.common.exceptions.InvalidSupportedTroubleTicketActions
import com.hp.uca.expert.vp.pd.interfaces.ActionsFactoriesSelection	import com.hp.uca.expert.vp.common.interfaces.ActionsFactoriesSelection
import com.hp.uca.expert.vp.pd.interfaces.SupportedActions	import com.hp.uca.expert.vp.common.interfaces.SupportedActions
Import com.hp.uca.expert.vp.pd.interfaces.SupportedTroubleTicketActions	import com.hp.uca.expert.vp.common.interfaces.SupportedTroubleTicketActions

2.1.1.2 What needs to be changed in your customized ProblemDefault

If you are overriding the following methods from ProblemDefault, they need to be changed because they do not exist anymore:

Method (in V3.1)	Should be replaced in V3.2 by
chooseSupportedActions(Alarm alarm, ProblemInterface problem)	chooseSupportedActions(Event event, CommonActionInterface problemOrPropagation)
chooseSupportedTroubleTicketActions(Alarm alarm, ProblemInterface problem)	chooseSupportedTroubleTicketActions(Event event, CommonActionInterface problemOrPropagation)

2.1.1.3 What needs to be changed in your customized ActionsFactory

If you are overriding the following methods from ActionsFactory, they need to be changed because they do not exist anymore:

Method (in V3.1)	Should be replaced in V3.2 by
createProblemAlarm(Action action, Scenario scenario, Group group, ProblemInterface problem, Alarm referenceAlarm)	createAlarm(Action action, Scenario scenario, GroupBase group, CommonActionInterface problemOrPropagation, Event referenceEvent)
terminateAlarm(Action action, Scenario scenario, Alarm alarm, ProblemInterface problem)	terminateAlarm(Action action, Scenario scenario, Alarm alarm, CommonActionInterface problemOrPropagation)

Method (in V3.1)	Should be replaced in V3.2 by
clearAlarm(Action action, Scenario scenario, Alarm alarm, ProblemInterface problem)	clearAlarm(Action action, Scenario scenario, Alarm alarm, CommonActionInterface problemOrPropagation)
acknowledgeAlarm(Action action, Scenario scenario, Alarm alarm, ProblemInterface problem)	acknowledgeAlarm(Action action, Scenario scenario, Alarm alarm, CommonActionInterface problemOrPropagation)
unacknowledgeAlarm(Action action, Scenario scenario, Alarm alarm, ProblemInterface problem)	unacknowledgeAlarm(Action action, Scenario scenario, Alarm alarm, CommonActionInterface problemOrPropagation)
associateAlarmsForHistoryNavigation(Action action, Scenario scenario, Group group, Collection Alarm children, ProblemInterface problem)	associateAlarmsForHistoryNavigation(Action action, Scenario scenario, GroupBase group, Collection Alarm children, CommonActionInterface problemOrPropagation)
dissociateAlarmsForHistoryNavigation(Action action, Scenario scenario, Group group, Collection Alarm children, ProblemInterface problem)	dissociateAlarmsForHistoryNavigation(Action action, Scenario scenario, GroupBase group, Collection Alarm children, CommonActionInterface problemOrPropagation)
setHistoryNavigation(Action action, Scenario scenario, Alarm alarm, Qualifier qualifier)	setHistoryNavigation(Action action, Scenario scenario, Alarm alarm, QualifierInterface qualifier)
setGenericAttribute(Action action, Scenario scenario, Alarm alarm, Command command)	setGenericAttribute(Action action, Scenario scenario, Alarm alarm, Command command)

2.1.1.4 What needs to be changed in your customized TroubleTicketActionsFactory

If you are overriding the following methods from TroubleTicketActionsFactory, they need to be changed because they do not exist anymore:

Method (in V3.1)	Should be replaced in V3.2 by
createTroubleTicket(Action action, Scenario scenario, Group group, ProblemInterface problem, Alarm referenceAlarm, List Alarm alarmsToAssociate)	createTroubleTicket(Action action, Scenario scenario, GroupBase group, CommonActionInterface problemOrPropagation, Alarm referenceAlarm, List Alarm alarmsToAssociate)
closeTroubleTicket(Action action, Scenario scenario, ProblemInterface problem, String troubleTicketIdentifer)	closeTroubleTicket(Action action, Scenario scenario, CommonActionInterface problemOrPropagation, String troubleTicketIdentifer)
associateTroubleTicket(Action action, Scenario scenario, Group group, ProblemInterface problem, List Alarm alarmsToAssociate, String troubleTicketIdentifer)	associateTroubleTicket(Action action, Scenario scenario, GroupBase group, CommonActionInterface problemOrPropagation, List Alarm alarmsToAssociate, String troubleTicketIdentifer)

Method (in V3.1)	Should be replaced in V3.2 by
`dissociateTroubleTicket(Action action, Scenario scenario, Group group, ProblemInterface problem, List Alarm alarmsToDissociate, String troubleTicketIdentifer)`	`dissociateTroubleTicket(Action action, Scenario scenario, GroupBase group, CommonActionInterface problemOrPropagation, List Alarm alarmsToDissociate, String troubleTicketIdentifer)`

2.1.2 In your XML configuration

Your ProblemXMLConfig.xml file (or equivalent) needs to be modified to make use of the new namespace "**http://config.im.vp.expert.uca.hp.com/**" for certain elements of the file like:

- actions
- troubleTicketActions
- booleans
- longs
- strings

You can use the ProblemXmlConfig-Migration-to-V32.xslt file part of the Inference Machine SDK to transform your current ProblemXmlConfig.xml version 3.1 to version 3.2.

Chapter 3

Fixed Problems

The following problems were fixed in this release.

Reference / Severity	Component	Description	Comment
CR#11894 High	Problem Detection VP	ProblemDefault never retrieves the delayForProblemAlarmClearance policy	Fixed
CR#12107 High	Problem Detection VP	Problem Detection does not correlate alarm on timewindow specified	Fixed
CR#11676 Medium	Problem Detection VP	Group_Alarm directive is not working when lower_Case is not set on TeMIP CA	Fixed
CR#11706 Medium	Problem Detection VP	User_Identifier of a Problem Alarm cannot be changed with calculateProblemAlarmOtherAttribute method	Fixed
CR#11710 Medium	Problem Detection VP	When an alarm is copied and cascaded, the local Variables are preserved, leading to potential serious issues	Fixed
CR#12600 Medium	Problem Detection VP	computeDelayForProblemAlarmCreation should be called by Problem Detection framework	Fixed
CR#12663 Medium	Problem Detection VP	the log supposed to say there is no supportedActions, is never sent ; instead a NPE is thrown	Fixed
CR#12801 Medium	Problem Detection VP	computeDelayForTroubleTicketCreation should be called by Problem Detection framework	Fixed

Table 2 - Fixed Problems in UCA EBC Problem Detection V3.2

Chapter 4

Known Problems

This section lists problems discovered during the product test campaign and that still have to be fixed:

Reference / Severity	Component	Description	Solution/Suggested workaround or comment
CR#11239 Medium	Problem Detection VP	UCA PbD: groupalarm does not work when <ToLower> not activated in temip CA	Possible User Error: On Hold. Will be addressed in a next release.
CR#10072 Medium	Problem Detection VP	Side effect of flag problemAlarmCanTriggerAnotherGroupForSameProblem on the parent field of cleared alarm	Workaround existed for our customer, but will need to be addressed.
CR#11917 Medium	Problem Detection VP	getTrigger() should not return ProblemAlarm (PA) when no more Trigger	On hold. Needs more investigation. Workaround exists.
CR#12958 Medium	Problem Detection VP	Resynchronization in IM is not complete	Will be addressed in a future release or patch.
CR#11061 Low	Problem Detection VP	when doing action, the same User name 'uca" should be used without any reference to the action id	Will be addressed in a future release.

Table 3 - Known Problems

Chapter 5

Known Limitations

No known limitation reported on the product yet.