HP Software

Unified Correlation Analyzer for Topology Based Correlation V1.2

TeMIP Integration Documentation

Edition: 1.0

For the HP-UX (11.31) and Linux (RHEL 5.2) Operating Systems

June 2011

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company

United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

Trademark Notices

Adobe® and Acrobat® 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 U.S. trademark of Sun Microsystems, Inc.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Netscape is a U.S. trademark of Netscape Communications Corporation.

NMOS™ is a trademark of RiverSoft Technologies Limited.

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

PostScript® is a trademark of Adobe Systems Incorporated.

UNIX® is a registered trademark of The Open Group.

Windows® and Windows NT® are U.S. registered trademarks of Microsoft Corporation.

X/Open® is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.

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.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Contents

Preface **7**

Chapter 1 Foreword **10**

Chapter 2 Main features **11**

Chapter 3 Global picture **12**

Chapter 4 Installation **13**

Chapter 5 Models and data-load **14**

Chapter 6 TeMIP Collector **15**

6.1 Role 15
6.1.1 Basic principles 15
6.1.2 Startup and resynchronization 15
6.1.3 Collection monitoring and retries 16
6.2 Basic Configuration 16
6.3 Running the TeMIP Collector 16
6.4 Advanced TeMIP Collector Configuration 17
6.4.1 TeMIP Collector property files 17
6.4.2 TeMIP Collector XML configuration file 19
6.4.3 Log4j file 25
6.5 TeMIP Collector Tools 25
6.5.1 runCollector 26
6.5.2 stopCollector 26
6.5.3 resyncCollector 26
6.5.4 sourceManager 26

Chapter 7 TeMIP Remote Handler **28**

7.1 Role 28
7.2 Basic Configuration 28
7.3 Running the TeMIP Remote handler 28
7.4 Call-outs 29
7.4.1 Raise Master alarm 29
7.4.2 Raise Root Cause Alarm 29
7.4.3 Update Root Cause Alarm 29
7.4.4 Clear Alarm 30
7.4.5 Update Alarm 30
7.5 Advance TeMIP Remote Handler Configuration 30
7.5.1 TeMIP Remote Handler generic part property file 30
7.5.2 TeMIP Remote Handler generic logging property file 31
7.5.3 TeMIP Remote Handler specific property file 32
7.5.4 TeMIP Remote Handler specifc Web service configuration file 32
7.5.5 TeMIP Remote Handler specific logging property file 33

7.6 UCA Alarm Object Custom fields 33
7.6.1 Remote Handler configuration 33
7.6.2 UCA Scenario Manager Configuration 34
7.6.3 TeMIP Client configuration 35

Chapter 8 TeMIP Service Manager OSS/J Trouble Ticket Support 36
8.1 Overview 36
8.2 Architecture 37
8.3 UCA Trouble Ticket Actions 38
8.3.1 Create TeMIP Trouble Ticket 38
8.3.2 Close TeMIP Trouble Ticket 38
8.3.3 Cancel TeMIP Trouble Ticket 39
8.4 Mapping Template Files 39
8.5 Basic Example 40
8.5.1 Model 40
8.5.2 Create a TT Associated to a Master Alarm 40
8.5.3 Clear all alarms and Close associated TT 41

Chapter 9 Problem detection example (hello world) 42
9.1 Description 42
9.2 Play this scenario step by step 43
9.2.1 Problem Detection example directory layout 43
9.2.2 Start UCA 45
9.2.3 Deploy the Problem Detection value-pack 47
9.2.4 Dataload instances into the UCA 48
9.2.5 Starting the engine 50
9.2.6 Check deployed rules 55
9.2.7 Load the test MSL 57
9.2.8 Create the demo Operation Context 58
9.2.9 Start the TeMIP-UCA integration processes 58
9.2.10 Simulate events 58
9.2.11 Navigate through correlated alarms 60

Chapter 10 Service impact and RCA example 62
10.1 Model 62
10.2 Transmission problem detection 63
10.3 Radio problem detection and Service Impact 64
10.4 Severity increase 65
10.5 Final picture 66

Glossary 67

Figures

Figure 1: interactions between components 12
Figure 2: Basic principles of TeMIP Collector 15
Figure 3: Update Alarm customized action dialog box 35
Figure 4: Example of UCA / OSS-J Integration 37
Figure 5: Trouble Ticket example, Meta Model 40
Figure 6: Create Trouble Ticket with associated alarms 40
Figure 7: Close Trouble Ticket 41
Figure 8: pattern detection: all BTS of a site are down 42
Figure 9: Desired output in TeMIP alarm handling window 43
Figure 10: ProblemDetection UML model 44
Figure 11: UCA Login page 46
Figure 12: UCA home page 46
Figure 13: UCA system manager window 47
Figure 14: UCA data-load window 48
Figure 15: Class/Instance file association 49
Figure 16: Import csv dialog 50
Figure 17: UCA status after startup 51
Figure 18: Updated data-load counters 52
Figure 19: adding a new demo user 53
Figure 20: UCA applications startup page 54
Figure 21: UCA's Mesh Viewer window 55
Figure 22: UCA's Scenario Manager window 56
Figure 23: Scenario manager with the ProblemDetection rules loaded 57
Figure 24: Mesh Viewer after the BTS alarms reception 59
Figure 25: Fired Rules Viewer after the BTS alarms reception 60
Figure 26: new SITE alarm created in TeMIP 60
Figure 27: Alarm navigation example 61
Figure 28: Service Impact example, Meta Model 62
Figure 29: Service Impact model, Instantiation 63
Figure 30: Transmission problem detection 63
Figure 31: New transmission alarm in TeMIP 64
Figure 32: Radio problem detection, and service impact up to UMT service 64
Figure 33: New Radio Problem and Service Impact alarms in TeMIP 65
Figure 34: Severity escalation on Service 65
Figure 34: Severity escalation in TeMIP 65
Figure 36: Service impact scenario, final picture 66
Figure 37: Service impact scenario, alarms correlated in TeMIP 66

Preface

Intended Audience

This document is aimed at the following personnel:

Delivery teams installing and using the TeMIP-UCA integration.

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:

The UCA software versions	TeMIP	UNIX	TeMIP Client
UCA V12I Level 0 **Revision A**	6.x	HP-UX Itanium (11.31)	TeMIP Client V6.2 Level 1 for Windows: • Windows XP • Windows Vista • Windows Server 2003
UCA V12L Level 0 **Revision A**	6.x	Red Hat Enterprise Linux Server release 5.2 (Tikanga)	TeMIP Client V6.2 Level 1 for Windows: • Windows XP • Windows Vista • Windows Server 2003

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 Topo TeMIP Integration Guide*
- *HP UCA Topo Installation and Configuration Guide*

- *HP UCA Topo TeMIP Client User Guide*
- *HP UCA Topo User Guide*
- *HP UCA Advanced Configuration and Troubleshooting Guide*
- *TeMIP-Service Manager OSSJ Trouble Ticket Liaison - User Guide*
- *TeMIP-Service Manager OSSJ Trouble Ticket Liaison – Installation & Configuration Guide*
- *TeMIP-Service Manager OSSJ Trouble Ticket Liaison - TeMIP Liaison Adapter System Integration Guide*

Support

Please visit our HP Software Web site at: www.hp.com/go/hpsoftwaresupport 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*
- *Updates*
- *Problem reporting*
- *Training information*
- *Support program information*

Terms and Acronyms

Table 1 - List of Terms and Acronyms

UCA	Unified Correlation Analyzer for Topology Based Correlation: new software for the NGOSS market, doing topology based alarm correlation and service impact
OC	TeMIP **O**peration **C**ontext
AO	TeMIP **A**larm **O**bject or UCA **A**ffected **O**bject
TeMIP Adapter	The adaptation software between UCA and TeMIP. Essentially composed of the Collector and Remote Handler applications (see beyond).
TWS	**T**eMIP **W**eb **S**ervice: the north-bound web-service interface to perform TeMIP calls
Call-out	The output of UCA. The call-outs are XML packets containing structured information. They are handled by the TeMIP Remote Handler, which map them to TeMIP directives.
Rule	The core of UCA is a JBoss rules engine.
Master Alarm	A new alarm created by UCA, usually grouping other

	TeMIP alarms together. UCA has the ability to group alarms together by creating a new one.
Contributory alarm/event	Alarms that contribute to a problem or service impact. They are the conditions, necessary and sufficient, to trigger the creation of a new master alarm.
Sympathetic alarm/event	Side effects of a problem or service impact. The resulting alarms are correlated to the master but are marked as "sympathetic" alarms. Often, the sympathetic alarms can arrive after the problem detection: they are therefore called "late arriving" sympathetic.
Correlation Tag	A new custom AO attribute, filled for master alarms created by UCA. Example values are "Service Impact", "Problem Report", "Root Cause Alarm".
(Association) Category	Or simply Category for short. It's a new column added in the Alarm Handling Window when the operator navigates from one alarm to the other. Its value is contextual regarding the source alarm. Example values are: "Contributory", "Sympathetic", "Master". The Category field qualifies an association between alarms and not an alarm. In particular, it is **not** an alarm attribute, even though it appears as such in the client window.
State Mesh	The internal network topology representation maintained in UCA. The mesh objects are loaded in database, and also in memory to maintain their state, and be visible to the rules engine.
Mesh object	One element of the state mesh.
Notification	Logical object used in UCA attached to a mesh object. A notification holds contributory and sympathetic events and can be associated to a newly created master alarm.
Instance Name	A mesh object attribute containing a TeMIP entity specification, making a correspondence with a TeMIP object.
Unique Reference	The unique identifier of a state mesh object in UCA.
RCA	**R**oot **C**ause **A**nalysis
SI	**S**ervice **I**mpact
PR	**P**roblem **R**eport
MSL	**M**anagement **S**pecification **L**anguage: the modelling language for TeMIP
FCL	**F**ramework **C**ommand **L**ine: the command line language of TeMIP
AHFM	**A**larm **H**andling **F**unctional **M**odule
ACS	**A**larm **C**ollection **S**ervice

Chapter 1 Foreword

This document is not the UCA user's guide nor an installation guide (these are available separately in the relevant documentation directory), but an overall description of the current integration between UCA and TeMIP.
It is assumed that you already have a significant knowledge of TeMIP and UCA to understand the wording (jargon) and general concepts used in this TeMIP integration description.

Since nothing replaces a simple use case to understand what a product can do, a basic "Problem Detection" sample scenario is proposed as a "hello world" step-by-step example, to eventually start with a concrete demonstration and see what UCA does live.

Chapter 2 Main features

UCA, which stands for Unified Correlation Analyzer for Topology Based Correlation, is a universal correlation engine, not specifically dedicated to TeMIP, which can be plugged to any management system to act as an external analyzer and service impact engine. However, it is currently tightly integrated with TeMIP to perform **topology-based** correlation and **service impact**. It can be seen as a replacement of TSM (TeMIP Service Monitor). It has also some **problem detection** or **root cause analysis (*)** abilities.

* We know that these words may have different meanings for various people, so we use them here in their intuitive sense. They will be defined later in more details, with examples.

UCA has no real operator user interfaces and can then be seen as a "black-box" engine. On the other hand, it has a rich development GUI and environment. When used for TeMIP, the output of the analysis is therefore sent back to TeMIP as **new alarms**, **alarm enrichment**, or the creation of **associations** between alarms, so that the results are directly visible in the Alarm Handling window of the TeMIP Client. For instance, new "service impact" alarms can be created in a dedicated Operation Context. Thus the number of alarms can be drastically reduced and only "root cause" or "service affecting" alarms can be displayed, with a drill-down facility to the other alarms, part of the problem but redundant.

As the AHFM does not yet manage associations between alarms natively, we have emulated this feature with additional AO attributes maintaining a list of "parents" and "children" alarms. Furthermore, a dedicated TeMIP client plug-in allows the operator to navigate through the resulting graphs or trees of alarms.

We don't reflect yet the calculated mesh states values in TeMIP or in a dedicated GUI (even though the mesh viewer GUI is available for rules developers).

The UCA integration makes an extensive use of the TeMIP Web Service (TWS) interface to TeMIP, which is therefore a necessary prerequisite for the integration.

UCA and TeMIP can be located on the same host or distributed. UCA itself can be made redundant (resilient) for fault tolerance and high availability.

Chapter 3 Global picture

The integration between TeMIP and UCA is composed of three software components: a "Collector", a "Remote Handler" and a TeMIP Client plug-in. The Collector and Remote Handler are two Java processes exchanging data between TeMIP and UCA through web-service interfaces. The Collector and Remote Handler applications are sometimes called the TeMIP "adapter" for UCA. The TeMIP Client plug-in displays alarm associations computed by UCA, and allows navigating through associated alarms (parent-child relationships).

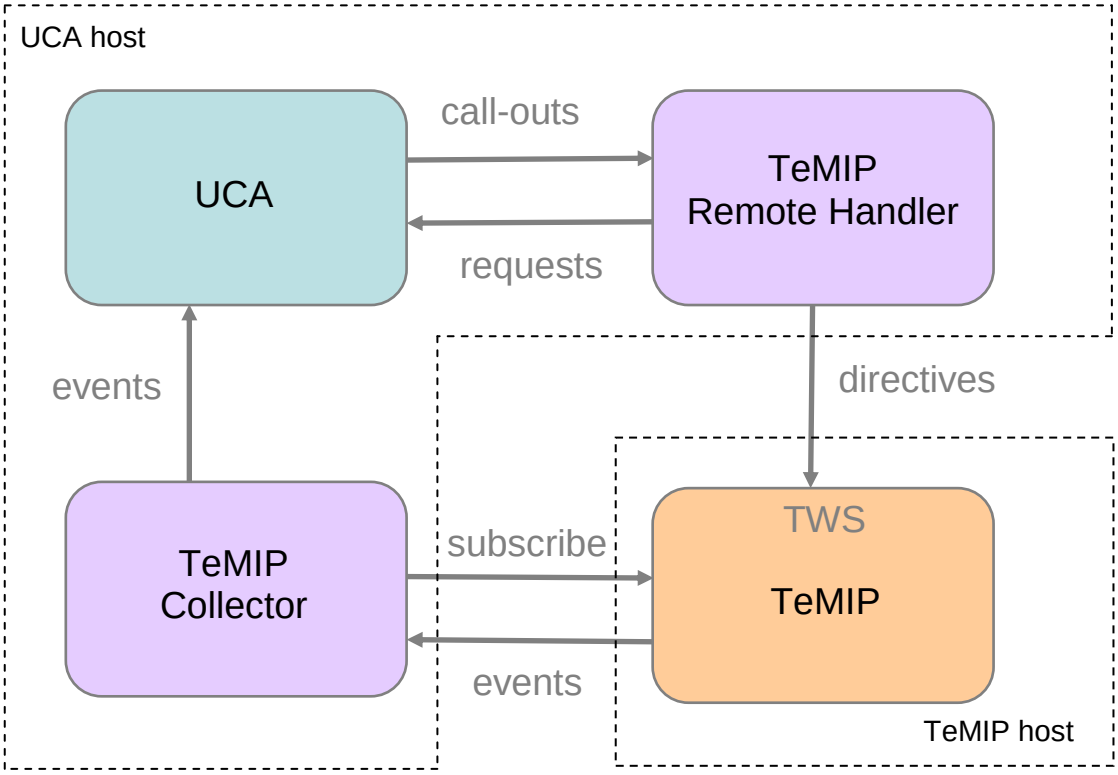

Figure 1: interactions between components

The TeMIP Collector is responsible for collecting alarms coming from some selected OCs, mapping them to the relevant XML format and forwarding them to UCA.
The TeMIP Remote Handler listens to the output of UCA, in the form of so called "call-outs" and maps them to TeMIP directives (mainly). The Remote Handler can also execute user defined scripts to perform any desired user defined actions.

Chapter 4 Installation

The TeMIP adapter is natively bundled with the UCA kit.
An automated setup procedure is available to complete the configuration after installation.
Two correlation “examples” are present in the kit as sample Value Packs (a set of files for rules, data-load, MSL and FCL scripts) demonstrating a simple use case. They are useful to test the correct behavior of the system but also (and essentially) to serve as example to see how to write rules, populate the state mesh, and use UCA.

The TeMIP Client plug-in is part of the TeMIP Client V6.2 Level 1 or upper version.

For the TeMIP-UCA integration to work, the TeMIP Web Services Northbound interface must be installed on the TeMIP side (at least on one director).

Currently, the Collector and Remote Handler processes must run on the **same** UCA host.

The following custom TeMIP Alarm Object attributes are added in the TeMIP dictionary for UCA:

10053 Correlation Tag
- 10054 UCA Notif Key
- 10061 UCA Custom Field1
- 10062 UCA Custom Field2
- 10063 UCA Custom Field3
- 10064 UCA Custom Field4
- 10065 UCA Custom Field5
- 10066 UCA Custom Field6
- 10067 UCA Custom Field7
- 10068 UCA Custom Field8
- 10069 UCA Custom Field9

Chapter 5 Models and data-load

There is no real constraint of model alignment between UCA and TeMIP: the two can work on fairly different, or on the contrary very similar, models.

In particular, no one-to-one mapping between TeMIP entities and UCA mesh objects is imposed. For instance, TeMIP can manage a typical network model made of managed objects and network equipment classes, while UCA can exhibit more abstract service or "domain logic" models, used for correlation and analysis.

UCA's mesh objects are loaded from files or native SQL commands (please refer to the UCA user's guide for more details). During the so-called "data-load" phase, the neighborhood of an object (i.e. the relationships with the sibling objects in the mesh) must be made explicit, and a unique reference name must be given to each object. In addition, the object can also be given an "alias" or so-called "instance name". The TeMIP integration makes use of this mesh object "instance name" attribute to hold a TeMIP entity specification, and therefore eventually link the UCA mesh object with a corresponding TeMIP instance. It is the responsibility of the integrator to populate theses "instance name" fields with the relevant TeMIP information during the UCA data-load phase. This entity specification value is then used by the Remote Handler when creating new alarms in TeMIP, to set the Managed Object mandatory attribute of the alarm. By default, if no "instance name" is given to the UCA mesh object, the TeMIP director name ("mcc 0") is used. This default value can be changed by configuration.

Here is example of a UCA data-load file, where the TeMIP entity specification is highlighted (the first line describes the file data-load columns format):

```
#Parent_Ref,Parent_Subclass,Parent_Class,Relative_Ref,Relative_Subcla
ss,Relative_Class,Class_Name,Subclass_Name,Instance_Name,Unique_Ref,S
ervice_State,Importance,Latitude,Longitude
Sprint 3 Model,V1.0,Model,ServiceComponent Sprint 3
Model,V1.0,Model,,,ServiceComponent,NetworkElement,Gateway,NetworkEle
ment .gateway_1,gateway_1_unique_ref,IN_SERVICE,1,0,0
```

So for instance, if one UCA rule raises an alarm on the "gateway_1_unique_ref" object, the Managed Object of the resulting alarm will be set to "NetworkElement .gateway_1". The Operation Context where to create the alarm is provided in the rule itself.

For more details on the CSV file format and the data-load phase, please refer to the UCA user guide.
On real projects, with big topologies and daily updates, UCA and TeMIP can be populated and synchronized with the help of the Unified Topology Manager tool (UTM). Please go to the UTM product documentation for more details.

Chapter 6 TeMIP Collector

6.1 Role

The TeMIP collector subscribes to a list TeMIP Operation Contexts (through the TeMIP Alarm Collection Server module) and transforms incoming AHFM configuration events, such as object creations or attributes value changes, into UCA events formatted in XML. The Subscribe call is made through the TeMIP Web-Service (TWS) interface. The TeMIP Collector runs on the UCA server hosts and is monitored (e.g started or stopped automatically) by the UCA server.

6.1.1 Basic principles

The TeMIP Collector is an alarm forwarder between TeMIP and the UCA server. It is bound on one side on the TeMIP ACS (Alarm Collection Server) via the TeMIP Web Services (TWS); and on the other side on the UCA server by using the Generic Collector API. It Processes TeMIP alarms from TeMIP to give them a format suitable for being processed by the UCA server as UCA events.

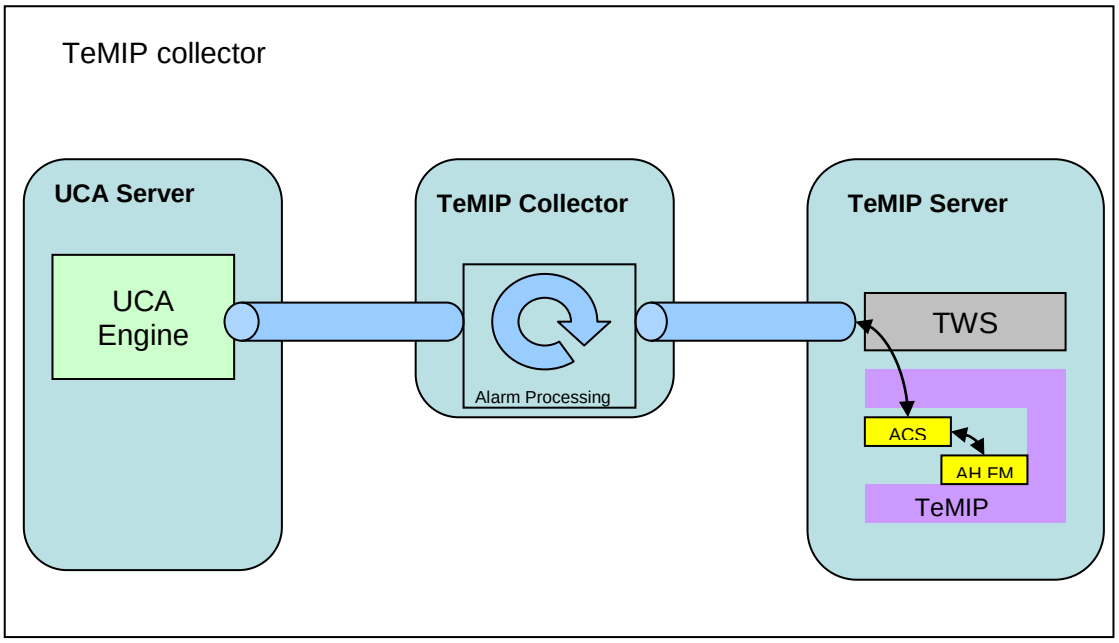

Figure 2: Basic principles of TeMIP Collector

6.1.2 Startup and resynchronization

When the TeMIP Collector starts, it performs a full re-synchronization before listening for normal incoming live event flow.
The resynchronization consists in collecting all pending alarms from the configured TeMIP Operation contexts (all alarms that are not terminated, by using the summarize

results of the Subscribe ACS directive) and sending them between two specific marker events (begin of synchronization/End of synchronization) to the UCA server.
On receipt of the resynchronization flow the UCA server reprocesses all the received events using a time contraction algorithm which ensures that the UCA server internal data (mesh and database) are in synch with the alarms states contained in the TeMIP Operation Contexts.
During this phase, the UCA server resynchronizes its event database with the TeMIP Operation Context alarm database. The associate rules are executed as if these alarm state changes (new alarm, state changes, termination) were received in a normal collection flow. The UCA server ensures that the rules execution do not lead to alarm duplication in TeMIP.

When the re-synchronization phase is over, the collector starts reporting the live events coming from normal (Getevent) ACS alarm collection.

6.1.3 Collection monitoring and retries

The TeMIP Collector is based on the TeMIP Web Service Client layer. This layer provides facilities for monitoring and - if required - re-establishing a collection to TeMIP when this collection has been interrupted
This is typically the case when the TEMIP application is stopped and restarted, but also if only parts of the TeMIP application are down for maintenance or for temporary unavailability (TWS, ACS, Alarm Handling).
In such case some retries are performed until the full collection chain is operational again. The collection is re-established and a full re-synchronization is performed again (as in the case of a TeMIP collector start).

6.2 Basic Configuration

The TeMIP Collector configuration is done through the UCA setup script that must be run after the installation (Refer to the "UCA installation guide" for that).
Another section "6.4 Advanced TeMIP Collector Configuration" gives full details on all configurations properties.

6.3 Running the TeMIP Collector

In a 'normal' production environment, the UCA server is capable of starting automatically the TeMIP collector and Remotehandler processes thanks to a set of rules called 'Resilience rules'. These rules also insure the event resynchronization and are performed automatically at UCA server startup.
However, in some specific cases (troubleshooting, development) one can have the need of starting the collector and remote handler manually.
This can be achieved by setting the following uca property (uca.properties file) as:

```
system.mode=standalone
```

and by using the following command to start the collector:

```
# $UCA_HOME/UCAcollector/bin/runCollector.sh
```

6.4 Advanced TeMIP Collector Configuration

This section gives the full details of all configurable parameters of the TeMIP collector. These parameters are located in two different configuration files. One is a properties file that drives the TEMIP collector behavior and its integration within the UCA system, the second is an XML configuration file dedicated to the TeMIP collection configuration.
The TeMIP Collector is controlled by 3 configuration files.

6.4.1 TeMIP Collector property files

Two property files are driving the TeMIP Collector configuration:

- One for the UCA Generic collection Part :

`$UCA_HOME/UCAcollector/configuration/ucacollector.properties`
(Which is a symbolic link to a file located in the /var/opt/uca directory) and contains the customizable variables driving the UCA generic event collector and specifically its connection with the UCA server.

All the properties defined in this file have default values set for a TeMIP collector running in a 'standard' configuration (i.e the remote handler is running on the same host than the UCA server, standard communication ports used).

`date.timezone`	The timezone in which the UCA systems operate. Default setting : GMT
`date.format`	The date/time format expected by the UCA DataCollector on both Server A & B. Default setting : yyyy-MM-dd HH:mm:ss
`genericcollector.hostname`	Should be set to the DNS name or IP address of the host on which the GenericCollector or its derivative is intended to run. Default setting : localhost
`managementservice.rmiport`	The RMI port number used by the UCA server Management service. Note: this should match the setting of the remoteHandler_TeMIP properties on each UCA server. Default setting : 18083
`managementservice.name`	The RMI name of the Management service. Note: this should match the setting of the remoteHandler_TeMIP properties on each UCA server. Default setting : Management_Service
`aserver.hostname`	The DNS Name or IP Address of the UCA A server platform. Default setting : localhost
`aserver.ipport`	The IP Port number used by the UCA A DataCollector Default setting : 6666
`bserver.hostname`	DNS Name or IP Address of the UCA B server platform. A value of 'none' means no server B present. Default setting : none

bserver.ipport	IP Port number used by the UCA B DataCollector Default setting : 6666
heartbeat.period	Number of seconds between two heartbeat messages to UCA servers. Default setting : 10
History.flushdivider	How many heart beats elapse between flushes of the notification history. Default setting: 0
socket.connectiontimeout	UCA DataCollector input socket timeout in milliseconds. Default setting : 250
autostart.rmiregistry	Flag controlling the rmiregistry auto-start. Default setting : true
webservice.username	The username for the UCA webservice endpoints. Default setting : system
webservice.password	The password for the UCA webservice endpoints. Default setting : system
datacollection.webservice	The name of the data collector's webservice on each peer. Default setting : datacollector/service
webservice.port	The port number of the data collectors webservice on each peer Default setting : 18080
buffer.size	The number of events that can be buffered waiting to send to UCA server sockets. Default setting : 1000
throttle.size	The number of events that can be buffered before the generic collector starts throttling input. Default setting : 100
throttle.sleep	How long the generic collector will throttle input for in milliseconds. Default setting: 10
secondary.resync.delay	Secondary resync delay, when sending a CYCLE_START there is the possibility of it racing a secondary resync start message. This property determines, in seconds, how long sendResyncCycleStart method waits for a secondary resync start to come in after sending a CYCLE_START. Default setting: 6

- One for the TeMIP Specific collection Part :
`$UCA_HOME/UCAcollector/configuration/temipcollector.properties`
(which is a symbolic link to a file located in the /var/opt/uca directory) and contains the customizable variables driving the TeMIP specific properties of the Collector.

`Collector_TeMIP.alarmTerminationPolicy`	Flag controlling the event termination policy. • A value of 'ClearAndTerminate' means that a termination event will be sent to the UCA server on both Clear Alarms and Terminated Alarms. • A Value of 'TerminateOnly' means that the terminate event will be forwarded only on receipt of terminated Alarm. Default Value: ClearAndTerminated
`uca.userDefAttr.notificationKey`	Alarm Custom attribute that will hold the UCA notification key. Default Setting: UCA_notif_key

6.4.2 TeMIP Collector XML configuration file

The file `$UCA_HOME/UCAcollector/configuration/TeMIP_configuration.xml` contains the customizable variables driving the Web service Client configuration for the connection to TeMIP. This file holds two different sections:

- The TeMIP Director Information
- The TeMIP Collection Information

6.4.2.1 TeMIP Director Information

```
<Authentication>
   <UserName>user</UserName>
   <Password></Password>
</Authentication>
<Axis>
   <RepositoryPath>conf/repository</RepositoryPath>
   <XmlPath>conf/axis2.xml</XmlPath>
</Axis>
<DirectorConfiguration>
   <MachineName>supra.fra.hp.com</MachineName>
   <TeMIPDirectorEntity>.temip.FM2_temip</TeMIPDirectorEntity>
   <TWSServerPort>7180</TWSServerPort>
</DirectorConfiguration>
<CallParameters>
   <BulkSize>150</BulkSize>
   <CallMaxDuration>5000</CallMaxDuration>
   <CallTimeOut>600000</CallTimeOut>
</CallParameters>
<EntityFiltering>
   <ToUpper>false</ToUpper>
   <ToLower>false</ToLower>
   <Trim>true</Trim>
   <FilterDot>true</FilterDot>
   <FilterDoubleQuote>true</FilterDoubleQuote>
</EntityFiltering>
```

<Authentication/>	Defines the authentication parameters to connect to the TeMIP Web Service North Bound Interface. Refer to the TWS User Documentation for authentication policies.
<Axis/>	Internal parameter, do not change it
<DirectorConfiguration/>	Defines the TeMIP Director information required to access the Web Service North Bound Interface
<CallParameters/>	Defines the parameters used for each TeMIP Call.
<EntityFiltering/>	Defines the way TeMIP instances name will be filtered on the whole application. For instance, depending on the follwing parameter, a TeMIP entity define in TeMIP as '.myinstance' can be used in UCA as 'MYINSTANCE'.

Authentication

<UserName/>	User used for all TeMIP calls
<Password/>	To ease the usage of the authentication, one basic security implementation is provided with the PWCallback class (See Note below). If you choose the "no Security" mode or if you customize the "OutflowSecurity" with a specific class (using a tier authentication tool for instance), this parameter is not required. Optional parameter.

Note

For authentication policies, refer to the TWS User Documentation and also the Security configuration file located at:

$UCA_HOME\collector_TeMIP\configuration\axis2.xml

Example of configuration in Low Security Mode:

```
<module ref="rampart" />

    <parameter name="OutflowSecurity">
        <action>
            <items>UsernameToken Timestamp</items>
            <passwordCallbackClass>com.hp.temip.temip_ws.common.p
        wcallback.PWCallback</passwordCallbackClass>
            <passwordType>PasswordText</passwordType>
        </action>
    </parameter>
```

The `com.hp.temip.temip_ws.common.pwcallback.PWCallback` class is a minimal security implementation using the User/Password parameters given in the TeMIP_configuration.xml file. Where

- The Username tag defines the UNIX user name
- The Password tag is the Unix password for this user.

One can customize or implement differently depending on its specific security constraints. In this case, it is not mandatory to specify the password in the configuration file.

Axis2

`<RepositoryPath/>`	`Internal parameter, do not change it`
`<XmlPath/>`	`Internal parameter, do not change it`

DirectorConfiguration

`<MachineName/>`	`The TeMIP director IP address or name.`
	`Example:`
	`16.133.155.256 or`
	`Machine.fra.hp.com`

`<TeMIPDirectorEntity/>`	`The name of the TeMIP Director`
	`Example:`
	`.temip.DIRECTOR1_director`

`<TWSServerPort/>`	`The port configured for the TeMIP Web Server North Interface.` `Example: 7180`

Note

For authentication policies, refer to the TWS User Documentation and This next section is related to the configuration of the TeMIP director that hosts the Web Services server.

The `MachineName` represents the TeMIP director IP address or name.

Example: 16.133.155.256 or Machine.fra.hp.com

The `TeMIPDirectorEntity` is the name of the TeMIP entity on that director

Example: .temip.ibis_temip

Note: the TeMIP entity name can be obtained by issuing the following command with an FCL_PM : “SHOW temip * “

The `TWSServerPor`t is the port configured for the TeMIP Web Server Interface (default is 7180).

Note that even if your TeMIP platform is distributed and has several directors, only one TWS access point may be defined. After this point the standard TeMIP call dispatching will be used.

CallParameters

<BulkSize/>	Passed to the Web server during a TeMIP call. Defines the maximum size of TeMIP Call reply packet during Web Service communication. Refer to TWS Documentation for more information. Example: 20
<CallMaxDuration/>	Passed to the Web server during a TeMIP call. It is the maximum time in millisecond before sending the bulk reply to the client, even if the bulk is not yet to its "BulkSize". Example: 5000
<CallTimeOut/>	Passed to the Web server during a TeMIP call. It is the time after one inactive call is removed from the server. Example: 600000

EntityFiltering

<ToUpper/>	Transform the TeMIP entity name to Upper Case in UCA.
<ToLower/>	Transform the TeMIP entity name to Lower Case in UCA.
<Trim/>	Trim the TeMIP entity name in UCA.
<FilterDot/>	Remove all dot '.' occurrence in the TeMIP entity name while transferring information to UCA.
<FilterDoubleQuote/>	Remove all Double Quote '"' occurrence in the TeMIP entity name while transferring information to UCA.

6.4.2.2 TeMIP Collection information

```
<OperationContexts>
   <OperationContext>temip_op</OperationContext>
</OperationContexts>
<CustomAttributes>
   <CustomAttribute>
      <Attribute>Custom Field1</Attribute>
      <Datatype>XmlString</Datatype>
   </CustomAttribute>
   <CustomAttribute>
      <Attribute>Parents</Attribute>
      <Datatype>XmlString</Datatype>
   </CustomAttribute>
</CustomAttributes>
<QueueSize>100</QueueSize>
<PassingClasses>
   <ClassHierarchy>
      <Class>BOX</Class>
   </ClassHierarchy>
```

```
</PassingClasses>
<DiscriminatorConstruct>
  <BlockingSubFilters>
    <BlockingSubFilter>
      <FilterItem>
        <attribute>Perceived Severity</attribute>
        <operator>equality</operator>
        <value>Indeterminate</value>
      </FilterItem>
    </BlockingSubFilter>
  </BlockingSubFilters>
  <PassingSubFilters>
    <PassingSubFilter>
      <FilterItem>
        <attribute>Additional Text</attribute>
        <operator>present</operator>
        <value></value>
      </FilterItem>
    </PassingSubFilter>
  </PassingSubFilters>
</DiscriminatorConstruct>
```

<OperationContexts/>	List of Operation Context <OperationContext/> subscribed in the Alarm Collection
<CustomAttributes/>	List of Customized Attributes that need to be decoded during Alarm reception
<QueueSize/>	Internal parameter, for a Message Queue. Example: 100
<PassingClasses/>	Optional parameter. List of <ClassHierarchy/> representing the entities that are effectively created/transferred to UCA. Internally, this parameter defines a piece of Discriminator Construct applied to Managed Object attribute. If not present, all entities and associated alarms are transferred to UCA.
<DiscriminatorConstruct/>	Optional parameter Defines the Discriminator Construct (Filter) parameter applied to all alarms coming through the TeMIP Service Console Collection. If not present, all alarms are transferred to UCA. Linked with the <PassingClasses/> tag. For additional information, refer to Alarm Filtering description in TeMIP Documentation.

OperationContexts

<OperationContext/>	An Operation Context name

CustomAttributes

<CustomAttribute/>	A Custom Attribute that needs to be decoded and transferred to UCA during Alarm reception.

CustomAttribute

<Attribute/>	The Attribute Name as described in the TeMIP metadata (TeMIP Dictionary Presentation name).
<Datatype/>	Datatype of the Attribute. Supported Datatypes: "XmlDecimal", "XmlString", "XmlBoolean", "EntitySpec", "EntitySet", "BinAbsTime"

PassingClasses

<ClassHierarchy/>	Sequence of <Class/> representing a path to a TeMIP class or subclass.

ClassHierarchy

<Class/>	The name of the TeMIP class

DiscriminatorConstruct

<BlockingSubFilters/>	Optional parameter List of <BlockingSubFilter/>. Alarms matching the criterias are not forwarded to UCA.
<PassingSubFilters/>	Optional parameter List of < PassingSubFilters />. Only alarms matching the criterias are forwarded to UCA.

BlockingSubFilters

<BlockingSubFilter/>	List of <FilterItem/>

BlockingSubFilter

<FilterItem/>	Defines the minimal information to specify a TeMIP filter

FilterItem

<attribute/>	The TeMIP Attributes that is subject to filtering as described in the TeMIP metadata (TeMIP Dictionary Presentation name). Example: 'Perceived Severity'
<operator/>	The operator used to evaluate Should be part of the list: "initialstring", "finalstring", "anystring", "present", "equality", "greaterOrEqual", "lessorEqual", "match" or "matchsyno"
<value/>	The value evaluated with the operator. Depends on the datatype of the attribute. Example: 'Indeterminate'

PassingSubFilters

<PassingSubFilter/>	List of <FilterItem/>

PassingSubFilter

<FilterItem/>	Defines the minimal information to specify a TeMIP filter

6.4.3 Log4j file

The file:
`$UCA_HOME/UCAcollector/configuration/log4j.properties`
is the standard Log4j configuration file for the TeMIP collector.
The syntax for log4j configuration is not given here, but can easily been found on the Internet.

6.5 TeMIP Collector Tools

This is a set of tools for TeMIPCollector administration purpose. These tools are used to start/stop the collector or make some dynamic configuration such as adding a new collection source (Operation Context) or making a re-synchronization of all active sources.

All these tools are based on JMX communication and as such need a JMX port to be specified. This port is the JMX port used by the TeMIPCollector which is by default 9999 but can be changed by positioning the `UCA_COLLECTOR_JMX_PORT` environment variable to another value before starting the TeMIPCollector (runCollector command).
All other tools must use the same value for a correct behavior.

All the TeMIP Collector tools are located under the `$UCA_HOME/UCAcollector/bin` directory.

`All the TeMIPCollector commands require the UCA_HOME environment variable to be defined on both Unix and Windows.`

6.5.1 runCollector

This is the command for starting the TeMIP collector.

Usage :

runCollector.sh

Description:
Starts the TeMIP Collector.

6.5.2 stopCollector

This is the command used to stop the TeMIP Collector.

Usage :
stopCollector.sh

Description:
Stop properly the TeMIP Collector. A stop request is sent to the TeMIP Collector via a JMX bean request. The Collector stops the Collection by cancelling the pending TeMIP Calls. This allows to properly releasing the TeMIP resources allocated by the TeMIP Alarm collection chain. The TeMIP Collector process exits when all collections are cancelled.

6.5.3 resyncCollector

This is the Command used to resynchronize the UCA server with the collection sources.

Usage :
resyncCollector.sh

Description:
By using this command, the TeMIPCollector fully re-initializes its collection sources. On the TeMIP side this means the collection is restarted (including the summarize operation). On the UCA server side, all the summarized alarms are sent back as 'resync' event, forcing the UCA server to make a full resynchronization.

6.5.4 sourceManager

This is the command used to dynamically add/remove a new TeMIP source (Operation Context) as collection source.

Usage :
sourceManager.sh -add|-remove *source_name*

Description:

This command allows adding or removing dynamically a TeMIP operation Context to the set of monitored Operation Context. When a new Operation Context is successfully added, the standard summarize operation is performed leading to a re-synchronization of the summarized alarms on the UCA server side.

Options:

-add : to add monitoring of an additional operation context
-remove : to remove monitoring of an operation context.
source_name : operation context name.

Warning: invoking this command doesn't update the Operation Context list in the TeMIPCollector configuration file.

Chapter 7 TeMIP Remote Handler

7.1 Role

The TeMIP Remote Handler listens to call-outs from the UCA server and maps them to TeMIP directives, for example for creating new high-level alarms (i.e. “master” alarms) or grouping alarms together (association).

7.2 Basic Configuration

As for the Collector, a default configuration is made by the UCA setup.sh script run after then installation.

This configuration is sufficient most of the times. However, if a finer configuration is needed, the TeMIP Remote Handler is controlled by 5 configuration files:

RH generic part property file: `$UCA_HOME/jars/configuration/remotehandler.properties`	This is the configuration file that controls the connection to the UCA server.
RH generic part logging property file: `$UCA_HOME/jars/configuration/remotehandler.logging.properties`	This is the logging configuration file for the generic (i.e common to all remote handlers) part of the TeMIP Remote Handler.
RH TeMIP specific property file: `$UCA_HOME/remoteHandler_TeMIP/configuration/temipremotehandler.properties`	This is the configuration file that controls all actions towards TeMIP.
RH TeMIP specific logging property file `$UCA_HOME/remoteHandler_TeMIP/configuration/log4j.properties`	This is the logging configuration file for the TeMIP specific part of the Remote handler
RH TeMIP specifc Web service configuration file: `$UCA_HOME/remoteHandler_TeMIP/configuration/TeMIP_configuration.xml`	This is the configuration file that configures the Web service Client connection to TeMIP

Another section “7.5 Advance TeMIP Remote Handler Configuration” gives full details on all configurations properties.

7.3 Running the TeMIP Remote handler

In a ‘normal’ production environment, the UCA server is capable of starting automatically the TeMIP collector and Remotehandler processes thanks to the ‘Resilience’ set of rules.
However, in some specific cases (troubleshooting, development) one can need to start the collector and remote handler manually.
This can be achieved by setting the following uca property (uca.properties file) as:

```
system.mode=standalone
```

and by using the following command to start the remoteHandler:

```
# $UCA_HOME/UCAremotehandler/bin/runRemoteHandler.sh
```

When started manually, it is strongly recommended to start the Remote Handler **before** the Collector in order to guarantee that no actions will be missed during the resynchronization phase.

7.4 Call-outs

The purpose of a UCA Remote Handler is to listen to call-outs and map them to TeMIP directives. They are essentially Alarm Object directives, to create new alarms, update alarms, or demote alarms below previously created ones.

This section describes what the TeMIP remote handler does for the main standard UCA call-outs. For the description of the call-outs themselves, please refer the UCA Remote Handler API documentation.

7.4.1 Raise Master alarm

This call-out results in the creation of a new Alarm in TeMIP. The Correlation Tag attribute is set to the UCA notification type value (retrieved with a request to UCA as not present in the call-out).
The contributory events and sympathetic events are associated to the newly created alarm. Their "category" attributes in the Children AO are marked respectively as contributory or the sympathetic.

This is the principal means of correlating alarms together. A new "complex event" is created to group a bunch of contributory events together.

7.4.2 Raise Root Cause Alarm

This call-out results in the creation of a new Alarm in TeMIP. The Correlation Tag attribute is set to the UCA notification type value given in the action dialog (retrieved with a request to UCA as not present in the call-out).
The contributory events, sympathetic events, and master alarms are associated to the newly created alarm. Their "category" attributes in the Children AO are marked respectively.
The Problem Report list is discarded.

This call-out is the result of the "Perform Root Cause Analysis" UCA action, which is a high value-added algorithm for doing topology based analysis.

7.4.3 Update Root Cause Alarm

The content of this call-out is similar to the "Raise Root Cause" alarm one.
No new alarm is created when receiving this call-out. Only new associated alarms are added to the previously created master alarm (with a previous "raise root cause alarm" call-out).

7.4.4 Clear Alarm

This call-out results in a Clear_Alarm directive on the given Alarm Id.

7.4.5 Update Alarm

These call-out results in one or several Set directives on the given Alarm Object attributes.
Even though TeMIP supports only an “overwrite” policy for the Set directive (except for the Operator Note attribute), the Remote Handler emulates the “prefix” or “append” policies for a finer control of alarm modifications.

AO User-defined attributes can also be populated to enrich the alarm with UCA information (please refer to the UCA Alarm Object Custom fieldschapter).

7.5 Advance TeMIP Remote Handler Configuration

This section gives the full details of all configurable parameters of the TeMIP Remote Handler. These parameters are located in two different configuration files. One is a properties file that drives the TEMIP remote Handler behavior and its integration within the UCA system, the second is an XML configuration file dedicated to the TeMIP Web Service configuration.

7.5.1 TeMIP Remote Handler generic part property file

The file:
`$UCA_HOME/jars/configuration/remotehandler.properties` is the configuration file that handles the configuration of the connection to the UCA server.
All the properties defined in this file have default values set for a remote handler running in a ‘standard’ configuration (i.e the remote handler is running on the same host than the UCA server, standard communication ports used).

Here is the detailed list of supported properties:

`verbose.reports`	Control flag for verbose reporting (true|false). Default setting : true
`eventmanager.webservice`	The full URI of the EventManager web service, including the host, port and name of the EventManager web service end-point. Default value: http://localhost:18080/eventmanager/service
`notificationmanager.webservice`	The full URI of the NotificationManager web service, including the host, port and name of the NotificationManager web service end-point. Default value: http://localhost:18080/notificationmanager/service

notificationuiserver.webservice	The full URI of the NotificationUIServer web service, including the host, port and name of the NotificationUIServer web service end-point. Default value: http://localhost:18080/notificationuiserver/service
rulesserver.webservice	The full URI of the RulesServer web service, including the host, port and name of the RulesServer web service end-point. Default value: http://localhost:18080/rulesserver/service
datacollector.webservice	The full URI of the DataCollection web service, including the host, port and name of the DataCollection web service end-point. Default value: http://localhost:18080/datacollector/service
scripts.directory	Path to scripts directory on the host the remote handler is running on. Default value: scripts.
management.rmiport	The HeartbeatResponse RMI registry port (usually 18083 but may be changed if conflicts occur on platform running the generic collector engine) Note: this is only used for resilient UCA configurations Default value: 18083
management.service	The HeartbeatResponse RMI service name provided by the generic collector engine. Note: this is only used for resilient UCA configurations Default value: Management_Service
buffer.size	The RMI call buffer size Default value: 10
throttle.size	The number of rmi commands that can be buffered before the remote handler starts Default values: 3
Throttle.sleep	How long the remote handler will throttle rmi commands in milliseconds Default value 10

All other properties within this file should not be changed.

7.5.2 TeMIP Remote Handler generic logging property file

The file :

`$UCA_HOME/jars/configuration/remotehandler.logging.properties`

is the standard Log4j configuration file for the TeMIP generic part of the TeMIP remoteHandler.

7.5.3 TeMIP Remote Handler specific property file

The file `$UCA_HOME/remoteHandler_TeMIP/configuration/temipremotehandler.properties` (which is a symbolic link to a file in the /var/opt/uca directory) contains the customizable variables driving the TeMIP Remote Handler.

All properties in this file are commented out, meaning that the system default values are used. To change one of these properties, remove the comment sign at the beginning of the line and set the new value.

List of supported properties driving the TeMIP configuration

temip.default_oc	Operation Context where new alarms are created. The Operation context is usually specified in the 'raise alarm' action but if it is missing this value will be used. Default value: oc
temip.default_mo	Default manage object of newly created alarms. The Managed Object is usually specified in loaded topology (Instance_Name attribute of a mesh object) but if it is missing this value will be used. Default value: mcc 0

List of supported properties driving the configuration for Trouble Ticket directives.

temip.tt.user	User name given in Trouble Ticket directives. Default value: temip.
temip.tt.server.name	TT Server name Global class TeMIP TT_SERVER of the JSR91_FM and represents the Trouble Ticket Server Default value:TT_SERVER SM
temip.tt.template.create	OSS-J Mapping Template file for the Create TT operation in Trouble Ticket. Default value: createTroubleTicketByValueRequest.xml
temip.tt.template.associate	OSS-J Mapping Template file for the Create TT operation in Trouble Ticket. Default value: setTroubleTicketByValueRequest.xml
temip.tt.template.dissociate	OSS-J Mapping Template file for the Create TT operation in Trouble Ticket. Default value: trySetTroubleTicketsByValuesRequest.xml
temip.tt.template.close	OSS-J Mapping Template file for the Create TT operation in Trouble Ticket. Default value: closeTroubleTicketByKeyRequest.xml
temip.tt.template.cancel	OSS-J Mapping Template file for the Create TT operation in Trouble Ticket. Default value: cancelTroubleTicketByKeyRequest.xml

7.5.4 TeMIP Remote Handler specifc Web service configuration file

The file

`$UCA_HOME/remoteHandler_TeMIP/configuration/TeMIP_configuration.xml` addresses the TeMIP Web Service Client configuration points. This file holds mainly the TeMIP Director Information required by the TeMIP Remote Handler.

This TeMIP web service client configuration file is usually the same than the one used for the TeMIP Collector. You can refer to the "Advanced TeMIP Collector Configuration" for more details.
Note that the Collection section is not required within this files because very specific to the collector.

7.5.5 TeMIP Remote Handler specific logging property file

The file :

`$UCA_HOME/remoteHandler_TeMIP/configuration/log4j.properties`

is the standard Log4j configuration file for the TeMIP specific part of the TeMIP remoteHandler.

7.6 UCA Alarm Object Custom fields

Nine TeMIP Alarm Object Fields have been reserved in the TeMIP dictionary for UCA users. They all have string values that can be set from rules through the "Update Alarm" action. Albeit the TeMIP MSL presentation name for these attribute is "UCA Custom Field X" in the TeMIP dictionary, the visible field name in the "Update Alarm" action dialog box can be easily be changed with configuration to a more meaningful name (e.g "temperature" or whatever) . The TeMIP Remote Handler is then responsible for mapping the UCA meaningful name to the corresponding generic TeMIP name.

The registered AO custom fields are the following:

10061 UCA Custom Field1
10062 UCA Custom Field2
10063 UCA Custom Field3
10064 UCA Custom Field4
10065 UCA Custom Field5
10066 UCA Custom Field6
10067 UCA Custom Field7
10068 UCA Custom Field8
10069 UCA Custom Field9

You can perfectly keep these default names in the actions. If you prefer using more meaningful names in your UCA rules, two configuration steps are necessary: customize the "Update Alarm" action dialog box, and customize the TeMIP Remote Handler.

7.6.1 Remote Handler configuration

The file TeMIP remote handler configuration file:
`$UCA_HOME/UCAremoteHandler/configuration/temipremotehandler.properties`
contains the following properties to define the AO custom fields aliases (they are all commented by default):

```
# temip.ao.ucacustomfield1.alias :
# temip.ao.ucacustomfield2.alias:
# temip.ao.ucacustomfield3.alias :
# temip.ao.ucacustomfield4.alias :
# temip.ao.ucacustomfield5.alias :
# temip.ao.ucacustomfield6.alias :
# temip.ao.ucacustomfield7.alias :
# temip.ao.ucacustomfield8.alias :
# temip.ao.ucacustomfield9.alias :
```

If you wish to give a friendly name to one of the free UCA custom AO field, provide its name to the corresponding property value. For example:

```
temip.ao.ucacustomfield1.alias : myUcaStatus
temip.ao.ucacustomfield1.alias : myUcaText
```

This is sufficient for the TeMIP Remote Handler to know how to map the incoming UCA event change notification (i.e "callout" result of the Update Alarm action) to the corresponding TeMIP AO attribute.

7.6.2 UCA Scenario Manager Configuration

To make the meaningful name visible in the UpdateAlarm dialog box you need to edit the "gui.fieldname" property in the file:
`$UCA_HOME/properties/actiondialogkey.properties`.
The value for this property is comma separated list of event field name. Just add your new user friendly name there. For instance:

```
gui.fieldname       : combobox,Event Field Name, true, eventRank,
systemClass, systemInstance, eventId, dataType, originatingTime,\
updateState,moClass,moInstance,severity,alarmType,probableCause,specificProb
lems,\
additionalText,additionalTextTag1,additionalTextTag2,additionalTextTag3,\
additionalTextTag4,additionalTextTag5,additionalTextTag6,\
primoEvento,ultimoEvento,nomoApparato,rete,descrAllarme,journal,campo2,\
myUcaStatus, myUcaText
```

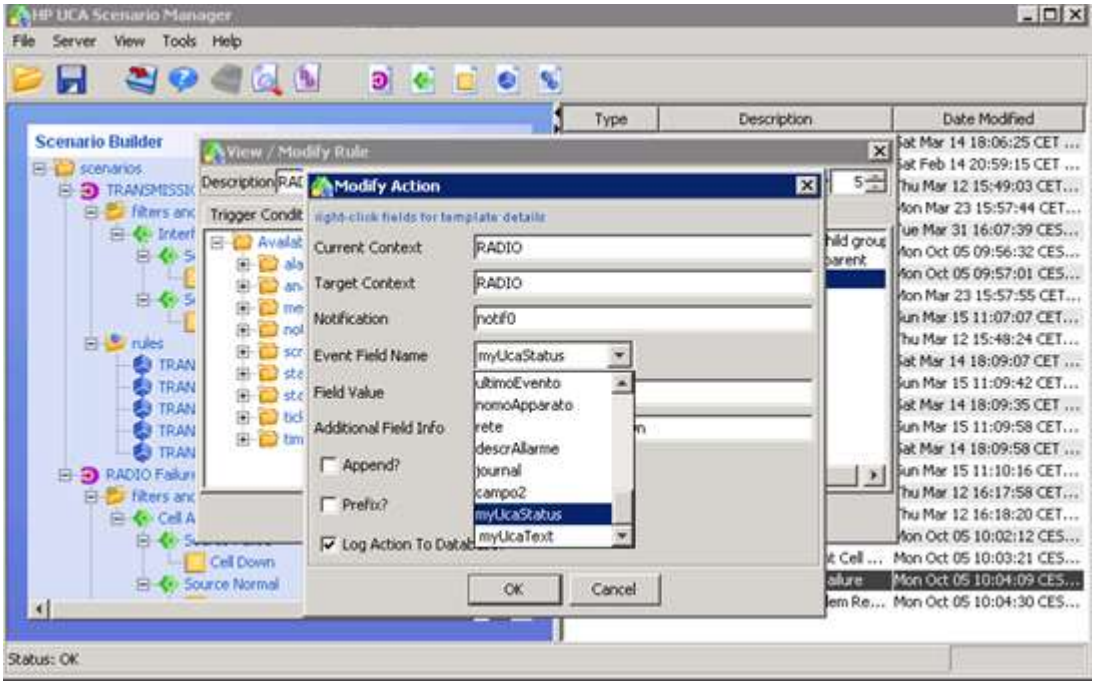

Figure 3: Update Alarm customized action dialog box

7.6.3 TeMIP Client configuration

Finally, it is also easy to customize the TeMIP Client Alarm Handling column names to reflect these new friendly names for the operator. Please refer to the TeMIP client documentation.

Chapter 8 TeMIP Service Manager OSS/J Trouble Ticket Support

8.1 Overview

The HP UCA / TeMIP-Service Manager OSS-J Trouble Ticket Liaison provides an end-to-end integrated service management solution in the trouble ticket domain based on a OSS/J JSR91 interface.

OSS/J JSR91 defines and standardizes a set of XML and Java APIs that facilitate the integration of OSS products with each other and makes it almost seamless. The OSS Trouble Ticket API is focused on defining a standard API that facilitates the data exchange among TT and non-TT components within the context of incident management.

One or more alarm objects in TeMIP can be associated with one or more trouble tickets through a case object, while trouble tickets can be mapped into incidents in HPSM. The TeMIP UCA-SM OSS/J Liaison manages these relationships, using the OSS/J JSR91 specification to communicate with these applications. It is based on 2 OSS/J Adapters:
The HP OSS/J Trouble Ticket Server Adapter for Service Manager
The HP OSS/J Trouble Ticket Client Adapter for TeMIP and UCA

Unified Correlation Analyzer can use the OSS/J JSR91 interface to perform Trouble Ticket Operations like:
Create Trouble Ticket
Close Trouble Ticket
Cancel Trouble Ticket

New UCA actions are available in the scenario designer user interface to integrate this trouble ticket rules in correlation scenarios.

8.2 Architecture

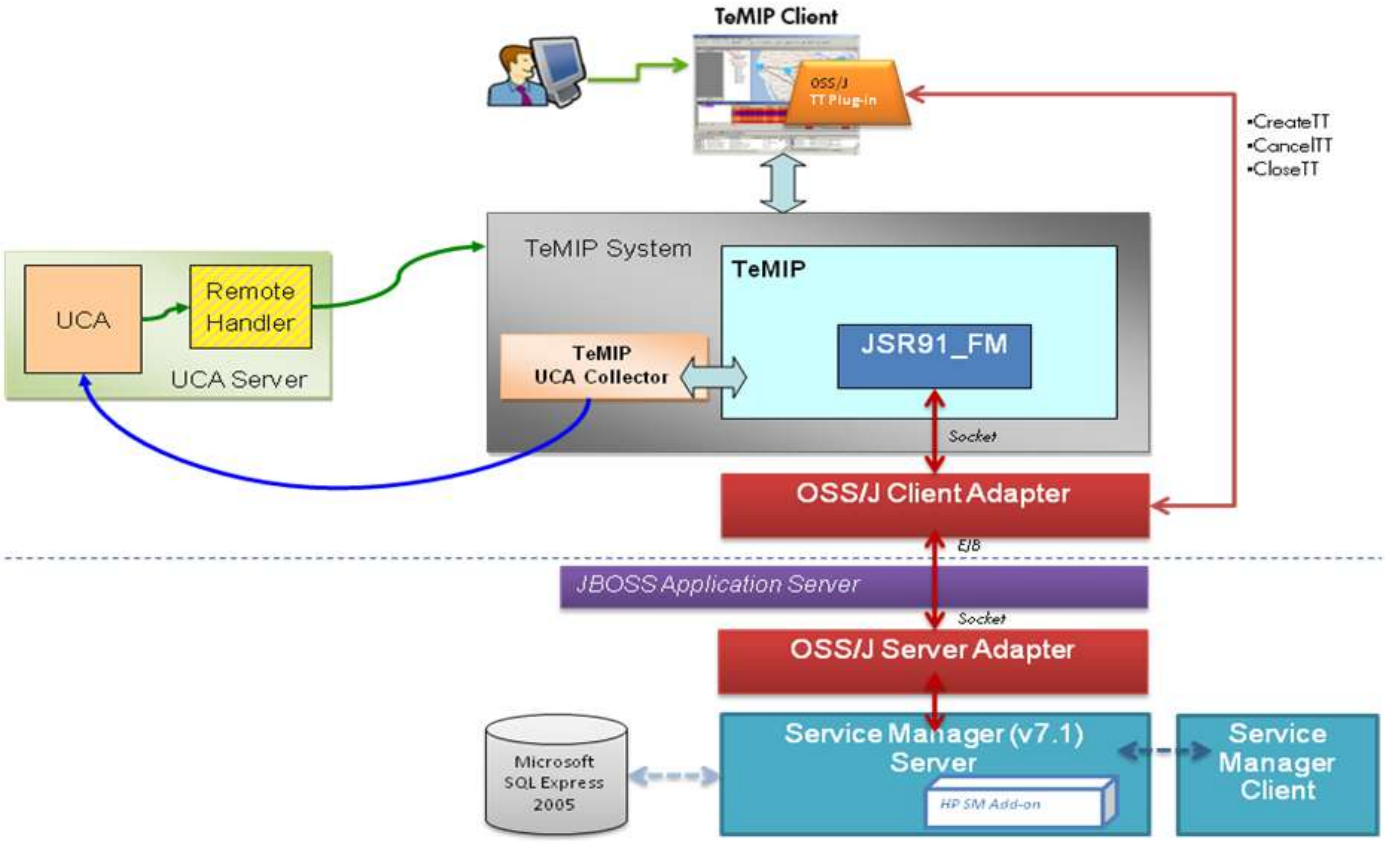

Figure 4: Example of UCA / OSS-J Integration

HP TeMIP is the Network Management Platform
HP UCA is the Unified Correlation Analyzer product in charge of the topology based correlation. It contains 2 parts: Collector in charge of collecting events from TeMIP and sending them to UCA correlation Engine, and the Remote Handler in charge of executing actions defined in the correlation rules.
HP JSR91 FM is the TeMIP Function Module, used to interface to TeMIP Server. (entity TT_SERVER)
HP OSS-J JSR91 Client Adapter is an adapter built to processes the JSR91 requests and TT Server notifications.
HP OSS-J Server Adapters is the JSR91 Adapter connection to the Application server where TT Server is deployed
HP Service Manager is the TT Server Manager that manages incidents.
HP TeMIP Client OSS-J Plug-in is the JSR91 Plug-in provides the user interface to the trouble ticket management operations. It interfaces between TeMIP Client and the JSR91 adapter through a socket communication, constructs a well-formed JSR91 request and sends it to the JSR91 adapter.

Note: OSS/J only supports today HP Service Manager as TT server.

UCA is only interface to the JSR91_FM directly to execute TT directives via the TeMIP Web Service interface to the TeMIP Entity TT_SERVER (global class of the JSR91_FM and represents the Trouble Ticket Server).

Please refer to the User documentation of HP SM OSS-J Trouble Tickets to have all the details on the OSS-J support in TeMIP.

8.3 UCA Trouble Ticket Actions

Specific Actions have been implemented to integrate OSS-J in UCA scenario designer.

8.3.1 Create TeMIP Trouble Ticket

This action creates a new case in the TT Server associating a list of alarms contributory and sympathetic.

Arguments	*Mandatory*	*Description*
Notification	Yes	UCA Notification identifier
Include contributory alarms	No	Checked if the contributory alarms are included in the Trouble ticket
Include sympathetic alarms	No	Checked if the sympathetic alarms are included in the Trouble ticket
Selected Alarms	Yes	The alarm list of alarms associated to the case. The first alarm is considered as the mapping alarm for the template file, and parents correlated alarms
Template File	No	Mapping Template file used for the create TT operation.. A default Template File will be used if this argument is empty (createTroubleTicketByValueRequest.xml)
User Input	Yes	This is an optional arguments dependant of the Mapping Template File. The user Input should be in the XML format.
Log Action to Database	No	This means that the Rule and associated action will be recorded in the UCA notification database

8.3.2 Close TeMIP Trouble Ticket

Arguments	*Mandatory*	*Description*
Notification	Yes	UCA Notification identifier
Template File	No	Mapping Template file used for the close TT operation.. A default Template File will be used if this argument is empty (closeTroubleTicketByKeyRequest.xml)
User Input	Yes	This is an optional arguments dependant of the

		Mapping Template File. The user Input should be in the XML format.
Log Action to Database	No	This means that the Rule and associated action will be recorded in the UCA notification database

8.3.3 Cancel TeMIP Trouble Ticket

Arguments	*Mandatory*	*Description*
Notification	Yes	UCA Notification identifier
Template File	No	Mapping Template file used for the close TT operation.. A default Template File will be used if this argument is empty (cancelTroubleTicketByKeyRequest.xml)
User Input	Yes	This is an optional arguments dependant of the Mapping Template File. The user Input should be in the XML format.
Log Action to Database	No	This means that the Rule and associated action will be recorded in the UCA notification database

8.4 Mapping Template Files

Mapping files are used to provide an efficient way to map alarm object information and incident case information (value association, function association or script association)
Mapping Template Files are in charge of the translation between the OSS-J Domain model and the HP SM incident model.
This Mapping enables operators to customize their mapping rules according to their business logic.

The OSS/J request template XML files are used to provide a default template when making the request. When making the request if user doesn't provide a value for a specific attribute, it will be filled in by the default value in the template.

The template files are located in the **/etc/hp/ism/adapters/jsr91adapter/templates** directory

It is recommended to read the complete documentation about Mapping in the OSS-J product to fully understand the feature. Any invalid template will fail a UCA trouble ticket operation.

8.5 Basic Example

8.5.1 Model

Imagine we have a simple model describing a site and 3 network equipments (cells). Each time a Master alarm is created by correlation rule, we want to automatically create a Trouble Ticket with the associated alarms hierarchy.

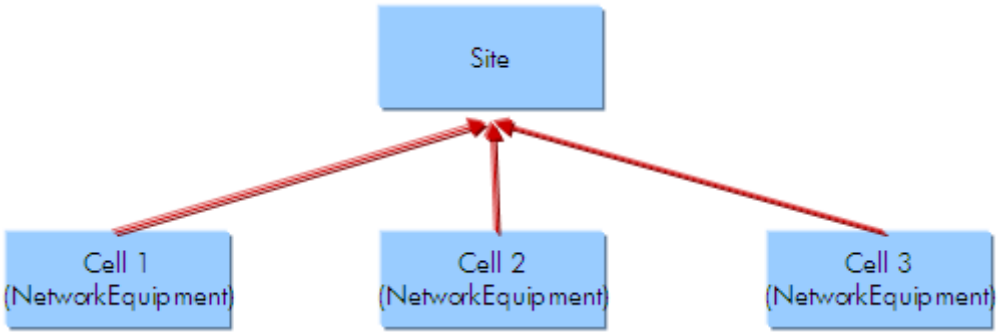

Figure 5: Trouble Ticket example, Meta Model

8.5.2 Create a TT Associated to a Master Alarm

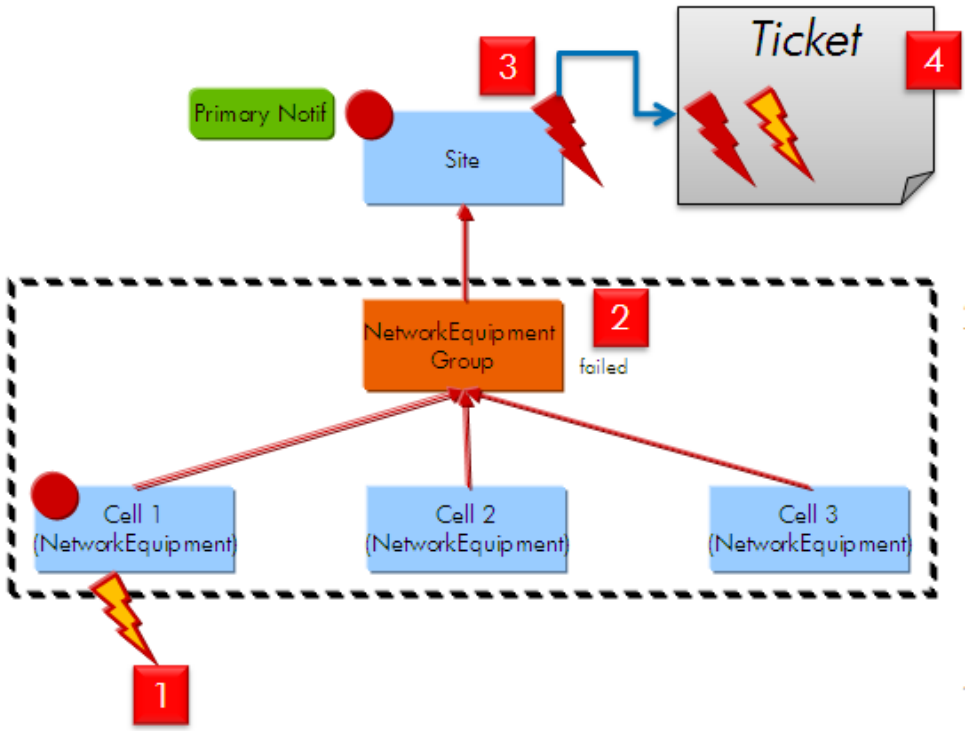

Figure 6: Create Trouble Ticket with associated alarms

1. Alarms Mapping rules generate state changes
2. Automatic Model-Driven State Propagation on NE Child Group

3. Rule-Driven Alarm creation: Raises an Problem Alarm on the Site when a Primary Notification is created and associate a contributory alarm (cell1)
4. Rule-Driven TT creation: Create a TT on the site If an Alarm exist and no Ticket Associated

8.5.3 Clear all alarms and Close associated TT

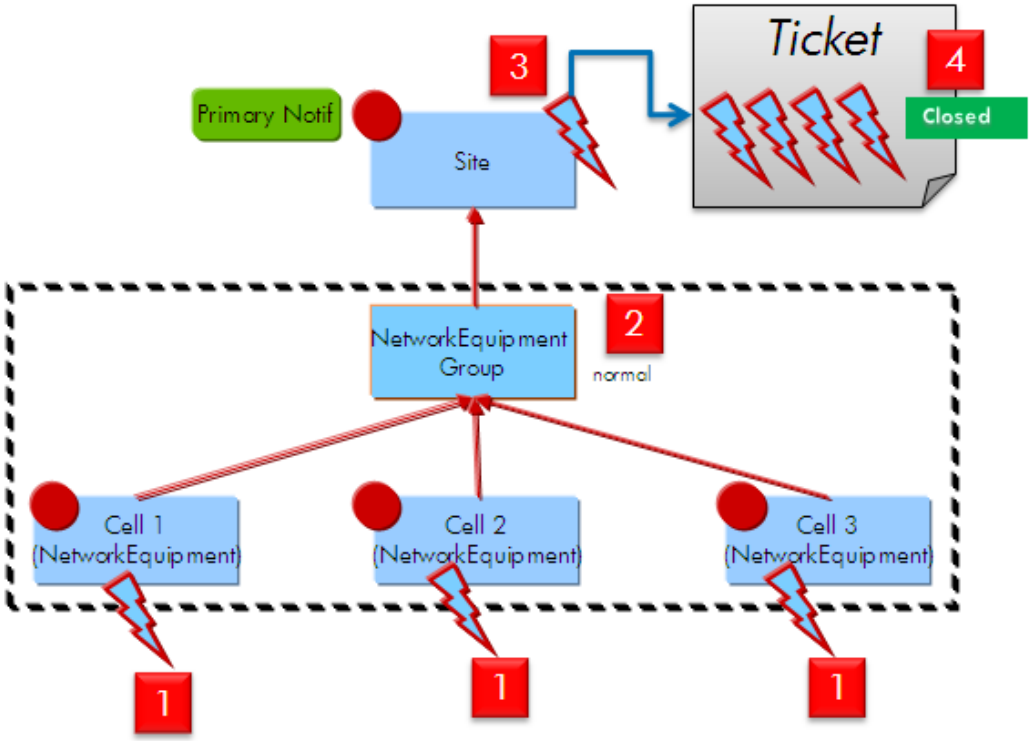

Figure 7: Close Trouble Ticket

1. Clearance received on all cells.
2. Automatic Model-Driven State Propagation on NE Child Group. Status back to normal.
3. Rule-Driven Alarm clearance: Master alarm is automatically cleared
4. Rule-Driven TT creation: Close TT if all associated alarms are cleared.

Chapter 9 Problem detection example (hello world)

9.1 Description

This example can be considered as a “hello world” scenario to start with UCA and test its effective integration with TeMIP. It is inspired from a very simplified GSM network management situation.

Imagine you have 3 base stations (BTS) on a GSM radio site (SITE). The BTS are managed by TeMIP and we receive alarms on them: indicating “BTS down”.

We then want to **detect** the situation where **all** BTS of the site are down, and create a new alarm for this “problem”. This is what we call a typical “problem detection” case: detect a **pattern** of alarms and make a single visible unit out of it, usually a new alarm grouping all the others.

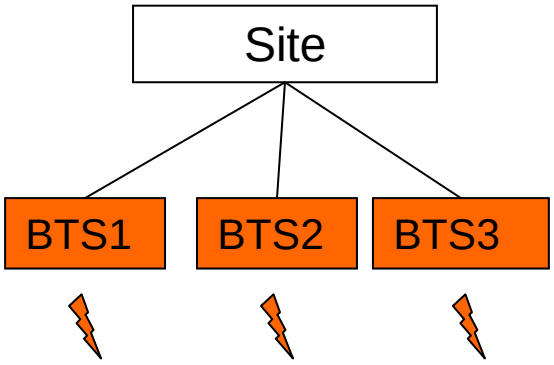

Figure 8: pattern detection: all BTS of a site are down

If we translate this in a fault management vision, we would like to create a new alarm representing this problem, and which group all the underlying alarms under it.

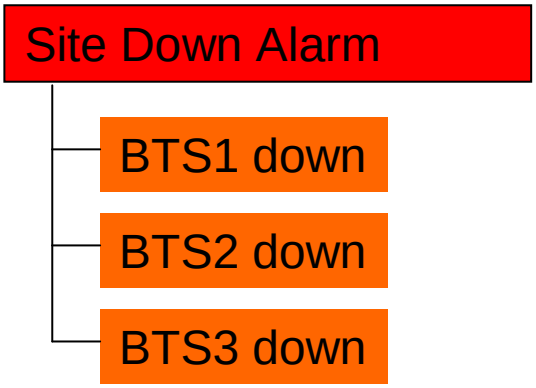

Figure 9: Desired output in TeMIP alarm handling window

Reversely, if one of the BTS alarm is cleared, the Site Alarm is then cleared automatically by UCA.

Note that this alarm can also be seen a "root cause alarm", for instance if the cause of the BTS down alarms is a power failure in the site.
New created alarms (master alarms), have a "Correlation Tag" attribute set to associated Notification type in UCA (refer to UCA user's guide) indicating for instance "Root Cause" or "Service Impact".

9.2 Play this scenario step by step

As a "hello world" example, we provide hereunder a detailed step by step procedure in order to run the scenario, and eventually discover UCA.
For an easier deployment/un-deployment of the complete example, this scenario is implemented as an UCA Value-pack. The following sections describe how to load the value pack, dataload instances for making a real test with TeMIP alarms.

9.2.1 Problem Detection example directory layout

As any other valuePacks, the Problem Detection example is delivered as a directory tree located under the $UCA_HOME/valuepacks directory.

The Problem Detection example directory hierarchy is made of a set of mandatory directories plus a set of directories containing data specific to this value-pack.

Mandatory files and directories:

vp-manifest.xml
models
actions
    rules

The **vp-manifest.xml** file is the Problem Detection value pack description file. It contains the name and description of the value-pack and also the name space to which it belongs.

The **models** directory contains the UML model that is loaded in UCA.
Two files are delivered :
ProblemDetection_model.xmi : The model is in the xmi format.
ProblemDetection_model.zargo : Editable format with the zargo editor

Note: if you are curious you may also have a look at the UML representation of the model by using the ArgoUML tool shipped with UCA, with a hyperlink at the bottom of the home page. It is a very basic generic representation of a telecom network. We use only a part of it in the scenario.

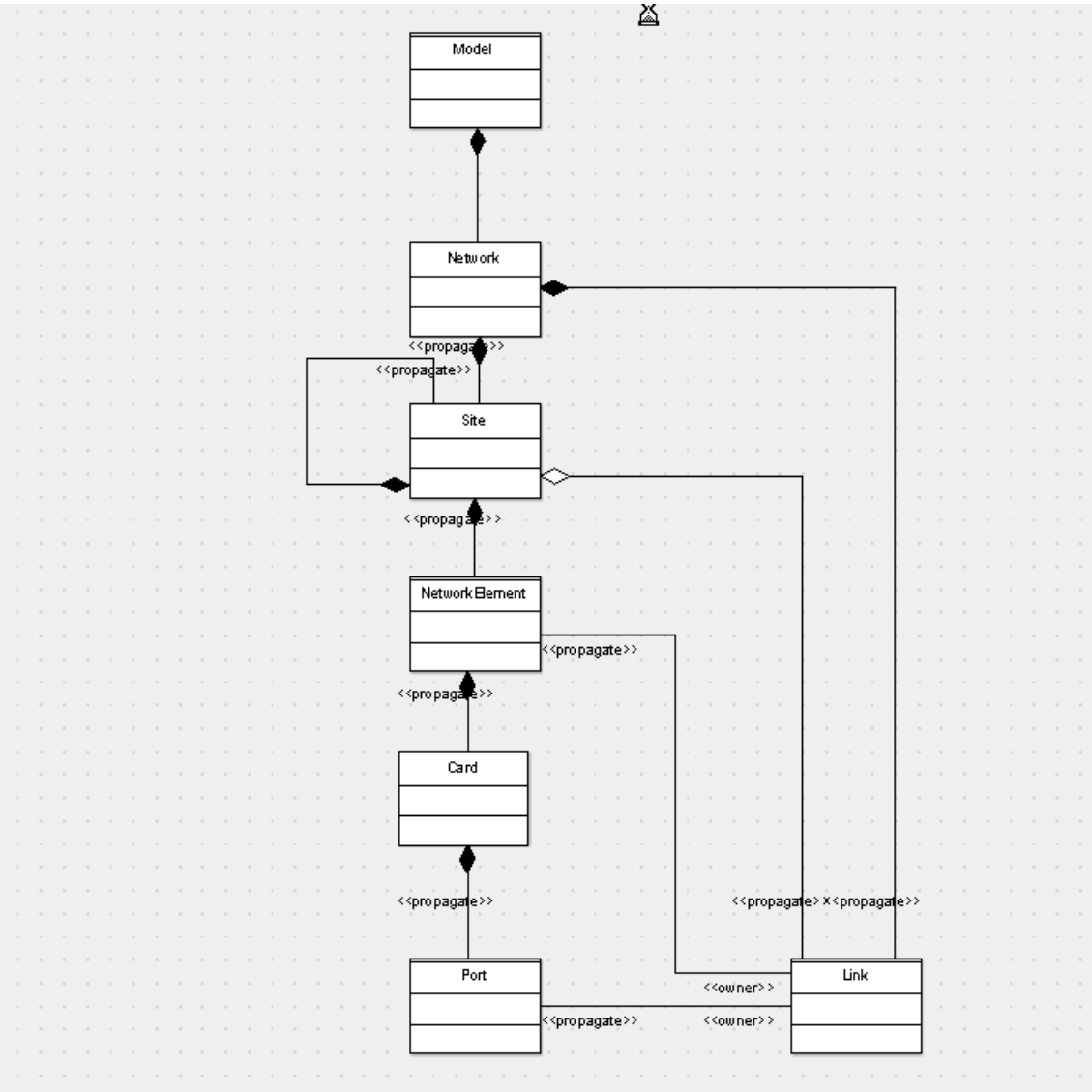

Figure 10: ProblemDetection UML model

This model is almost generic and could be applied to many network topologies. It is mainly based on containment relations (the black diamonds in the picture above indicate a UML composition relationship). A network is made of Sites that contain Network Element, that contain Cards, etc…

The **actions** directory is a mandatory and contains the specific actions that may have been developed for this value-pack. In the case of Problem Detection the rules are base on the standard system actions and thus this directory remains empty.

The **rules** directory contains the rules that implement the Problem detection scenario and that have to be loaded in UCA.

Specific files and directories:

The **dataload** directory contains the csv files used to populate UCA's topology mesh. Each file corresponds to a model class.

The **fcl** directory contains TeMIP scripts to simulate the incoming events to be able to play the scenario.

The **msl** directory contains the TeMIP part of the model, as a set of MSL files to be loaded. These new classes are needed in the TeMIP dictionary essentially because new alarms are created by UCA with a Managed Object that has to be present.

In the following sections, the file names are given with a path relative to this current ProblemDetection value-pack directory.

9.2.2 Start UCA

UCA is embedded in a tomcat server that should be started first:

```
# su - uca
# $UCA_HOME/bin/tomcatserver.sh
```

Note: it's important to log first as the uca user so that the UCA_HOME, JAVA_HOME and CATALINA_HOME environment variables are correctly set.

UCA's graphical user interfaces are all web based. Once the tomcat server has been started, you can open the UCA login page at the following URL:

http://<uca host name>:18080/uca/

Note: the port number depends on your configuration, the default value is given here. Also, make sure that the host name is reachable from your client host (where the browser is executing). Usually, the server name must be fully qualified with a domain name. You can also give directly the IP address instead if you know it!

Figure 11: UCA Login page

From the UCA login page, Use system/system as login/password to log in.
A successful login will show you the UCA home page:

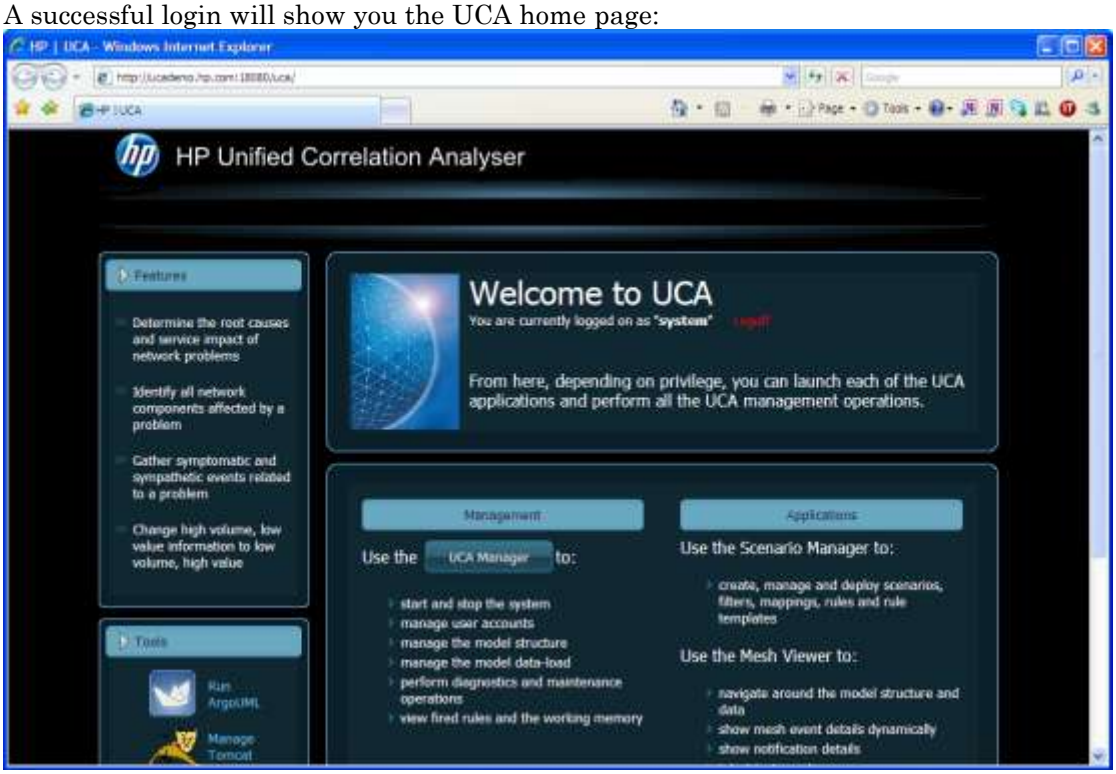

Figure 12: UCA home page

From the UCA home page, press the "UCA Manager" button to launch the System Manager applet window. Use system/system as login/password to log in and push the login button.

Note: The System Manager application is executed thanks to the "Java Web Start" utility that should therefore be installed on the client system. This a prerequisite for using UCA.

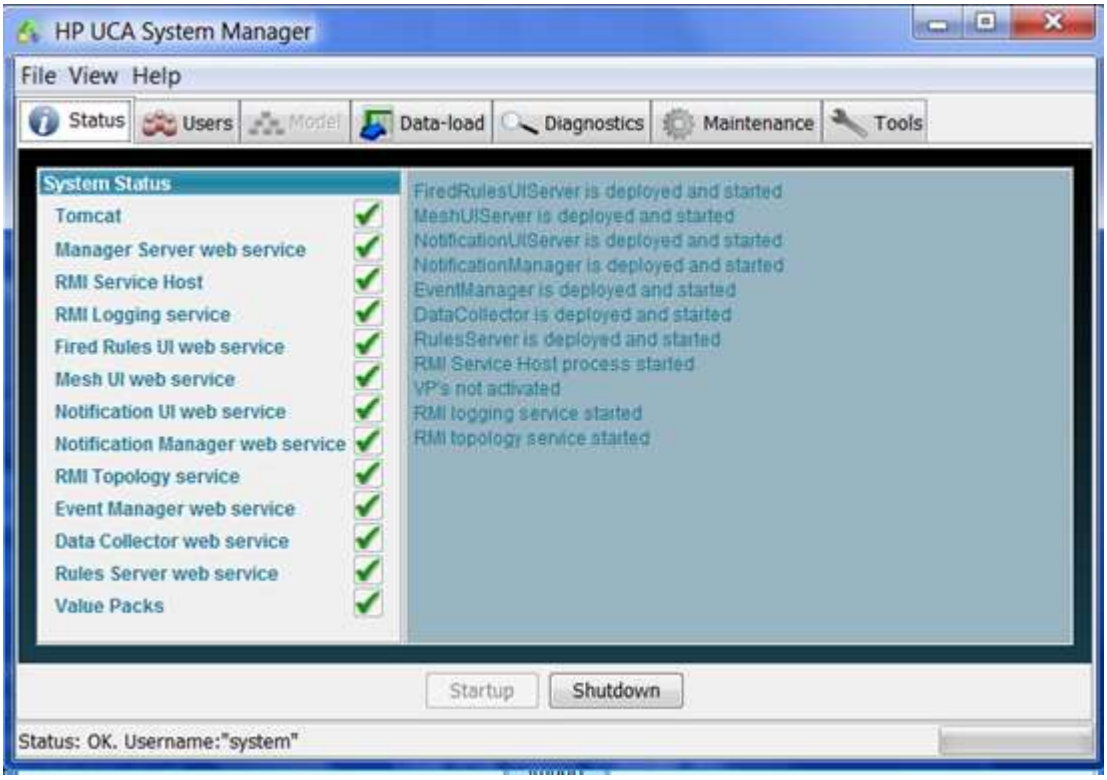

Figure 13: UCA system manager window

9.2.3 Deploy the Problem Detection value-pack

The Problem detection example deployment in UCA is made by using the vp_deploy.sh command line tool.

As the "uca" user, execute the following command:

```
# vp_deploy.sh hot-deploy ProblemDetection system system
VP deployed ok
#
```

By doing so, both the Problem detection model and rules are deployed in UCA server.

The effective deployment can be check with the following command:

```
# vp_deploy.sh list system system
```

9.2.4 Dataload instances into the UCA

UCA instances (objects corresponding to the UCA model) are loaded through the GUI before starting up the engine. In the System Manager, select the "Data-load" tab.

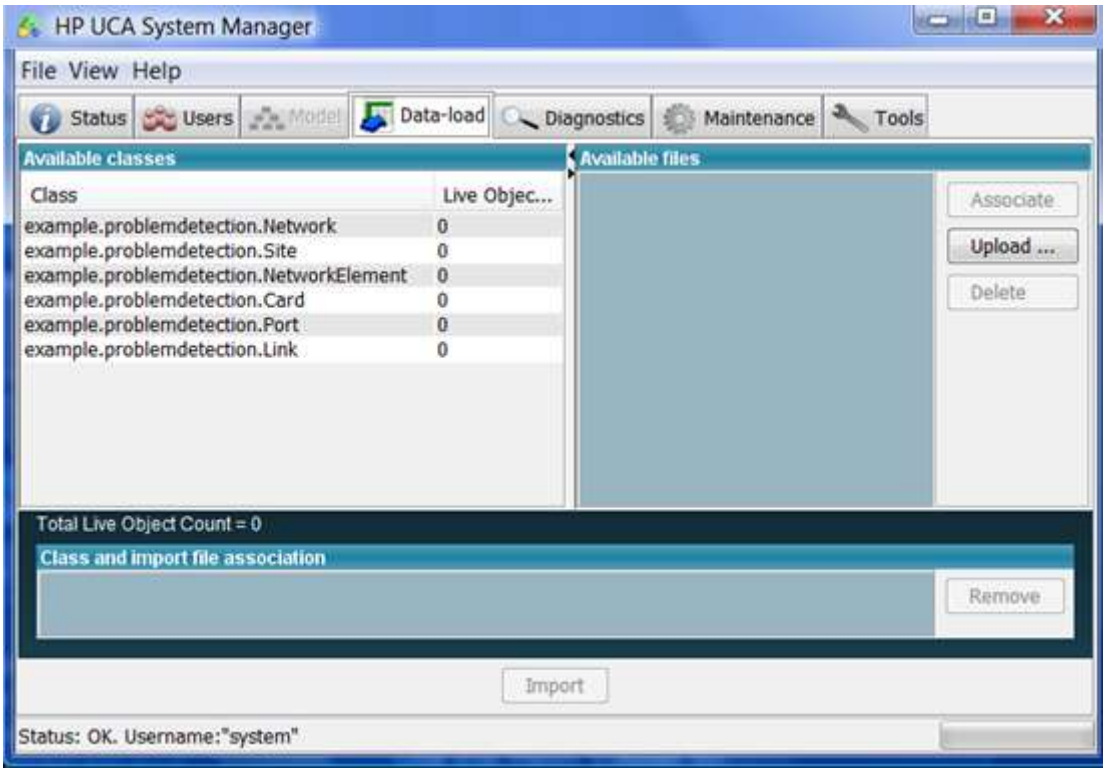

Figure 14: UCA data-load window

The left pane shows the count of objects currently loaded for the various classes in the Model.

Instances are organized by classes and loaded through comma-separated values files (CSV), to be found in the ProblemDetection/dataload directory.

Press the "Upload" button to add a file in the library. Do this for the following classes:

```
Network.csv
      NetworkElement.csv
      Site.csv
```

Then we need to associate one file to its corresponding class.
In the "Available classes" list select the "Network" line by clicking on it: it will remain highlighted. Then select "Network.csv" in the "Available files" list (which has been filled by the previous upload phase).
Once both lines are highlighted, press the "Associate" button.
You should see the "Class and Import File Association" section updated with the new association.
Not that no data-load has been achieved yet (until you press "Import") and that the counters are still at 0.

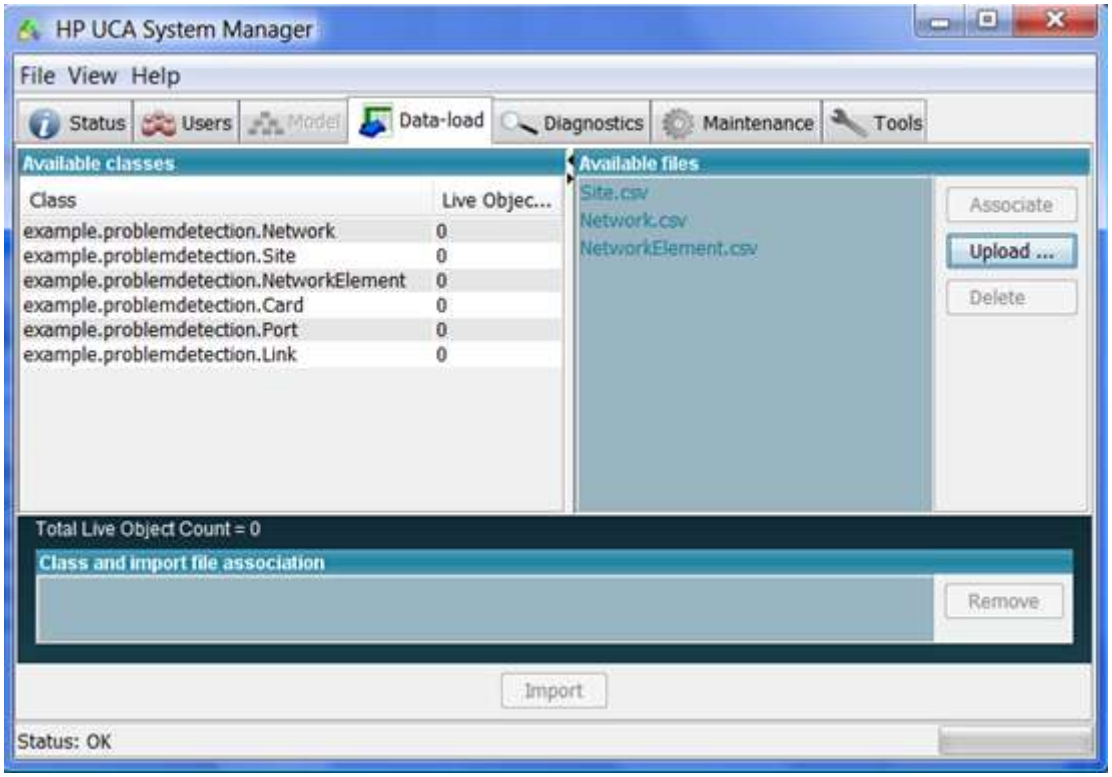

Figure 15: Class/Instance file association

Repeat the operation on the `NetworkElement.csv` and `Site.csv` files (logically associated with the NetworkElement and Site classes respectively).

<u>NOTE</u>: For the next sequence, UCA must be shutdown to have access to the models and Data-Load tabs. To stop the engine, go in the "Status" tab and press the "Shutdown" button. Once the data-load is complete, press "Startup". This sequence is different from the "uca_start" and "uca_stop" utilities, which also stops and starts the tomcat server.

Now, return back to the dataload tab and press the "Import" button to perform the data-load.

When prompted, make sure to check the "Delete the table(s) contents before importing" and "The CSV file(s) have a first-line header row" options before pressing "OK".

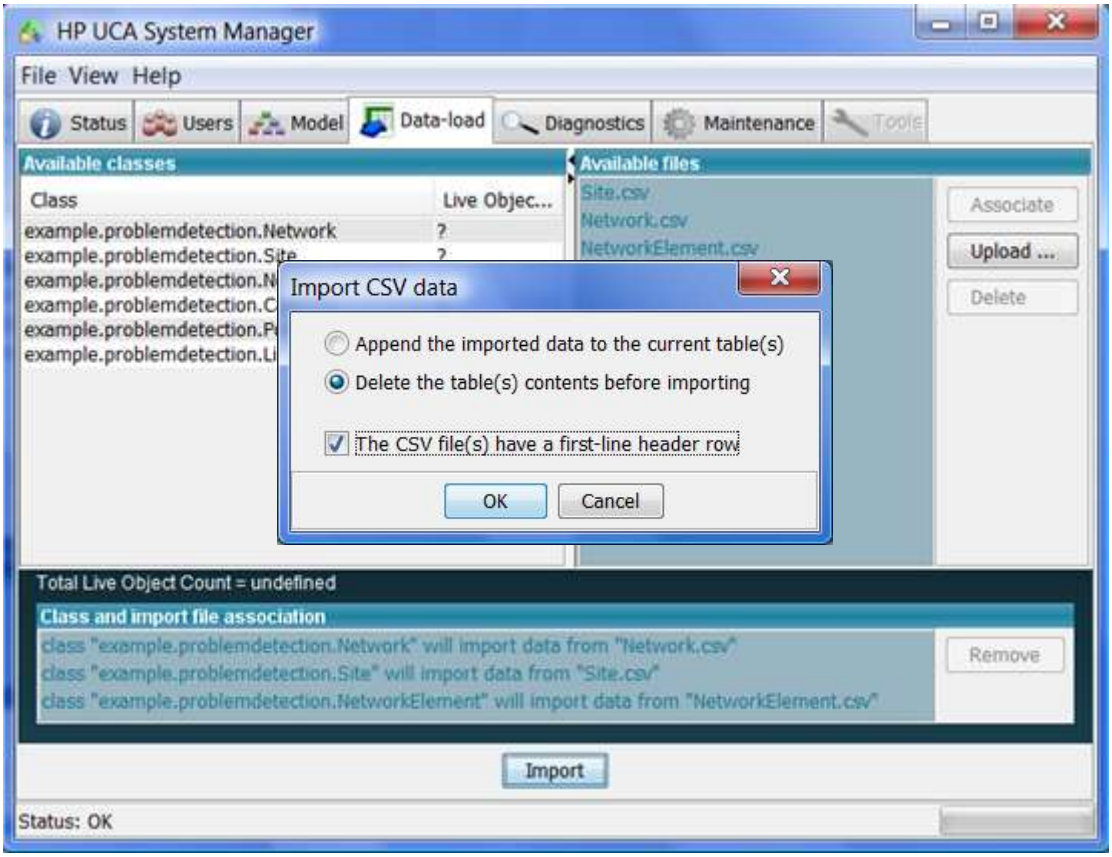

Figure 16: Import csv dialog

If the parsing of the files is correct, the import is silent. Note that the object counters will remain to 0 as long as we don't start the system, which is the next section!

9.2.5 Starting the engine

In the System Manager window, select now the "Status" tab, and press the "Startup" button. All UCA components statuses should go green.

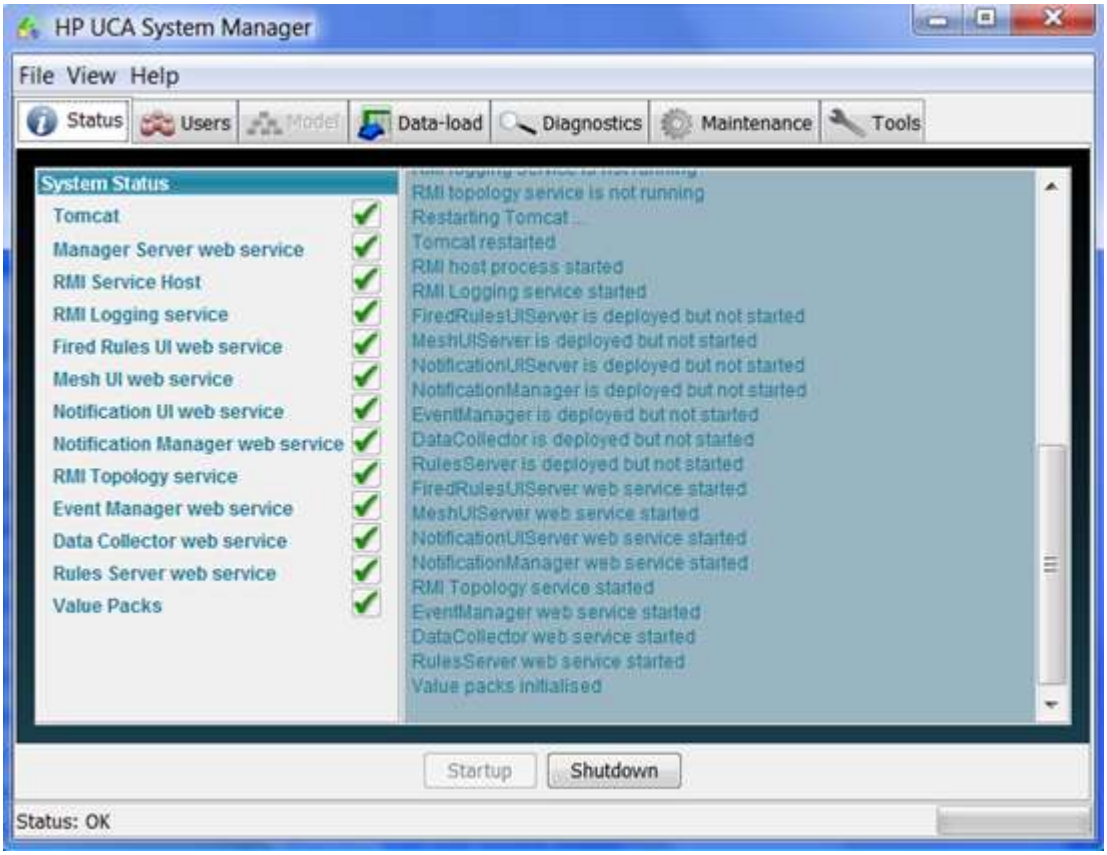

Figure 17: UCA status after startup

And if you go back to the "Data-load" tab, you should notice that the counters have changed.

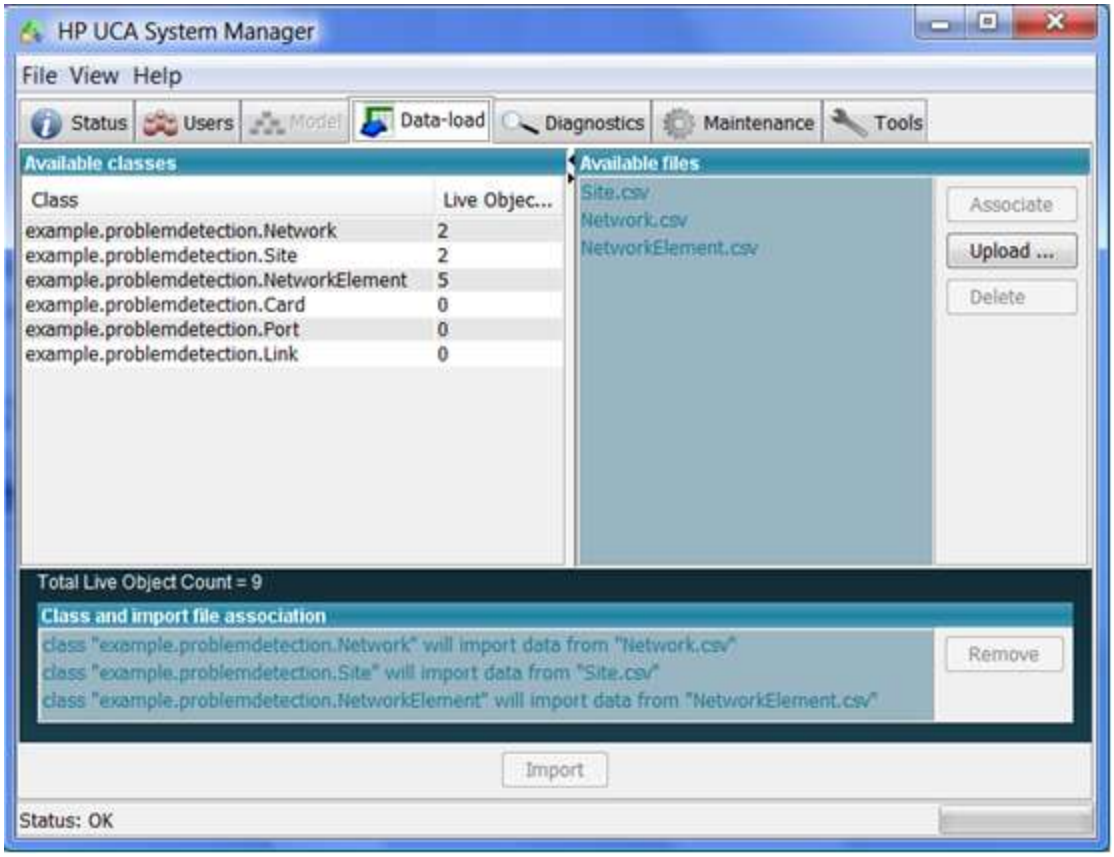

Figure 18: Updated data-load counters

Once the system has been started, we can now browse the state mesh to visualize the current states of the objects and deploy new rules to implement correlation scenarios. We do this be using applications called the "Mesh Viewer" and the "Scenario Manager".

We suggest that you create a new user in the UCA system, with the good credentials to use the applications for this demo. To do this, simply choose the "Users" tab in the System Manager and fill in the form as shown.

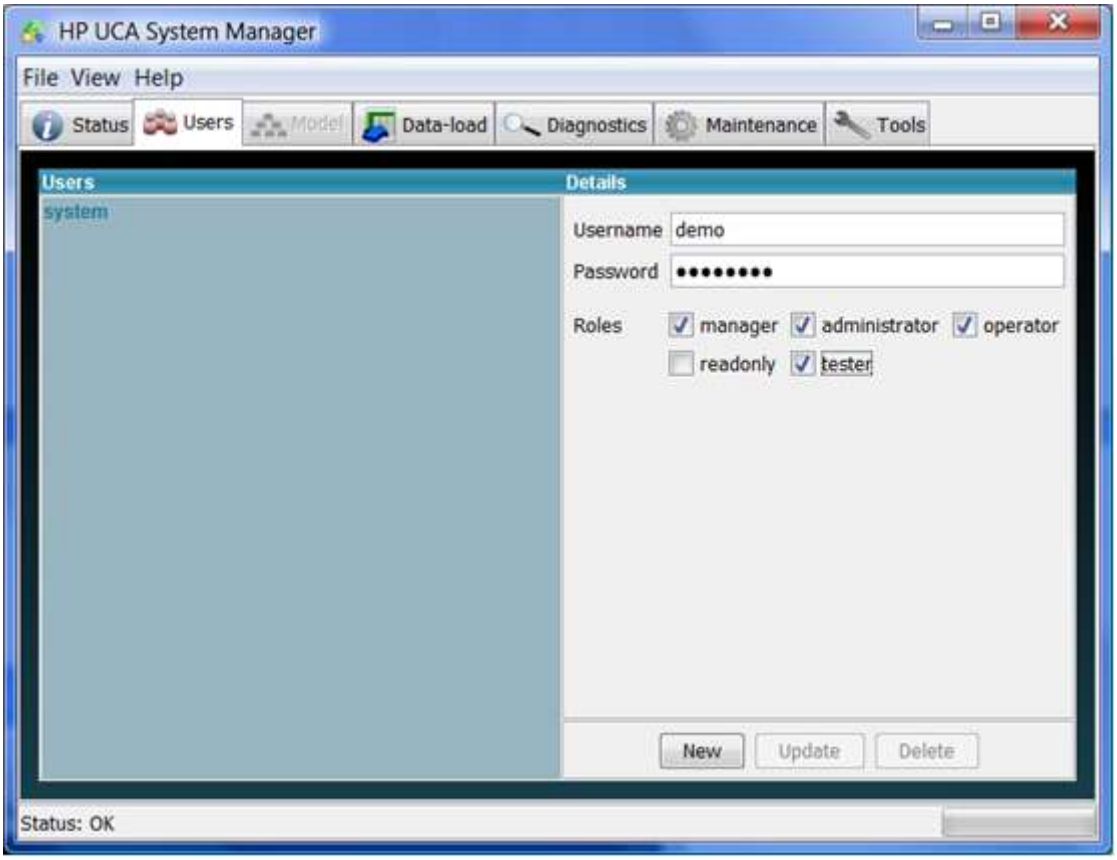

Figure 19: adding a new demo user

Now, log in the applications page with your new demo user (or else): open again the UCA home page (URL: http://<uca host name>:18080/uca/). Use the credentials of the newly created user.
And you should now see:

Figure 20: UCA applications startup page

Press the "Mesh Viewer" button to launch the Mesh Viewer applet.

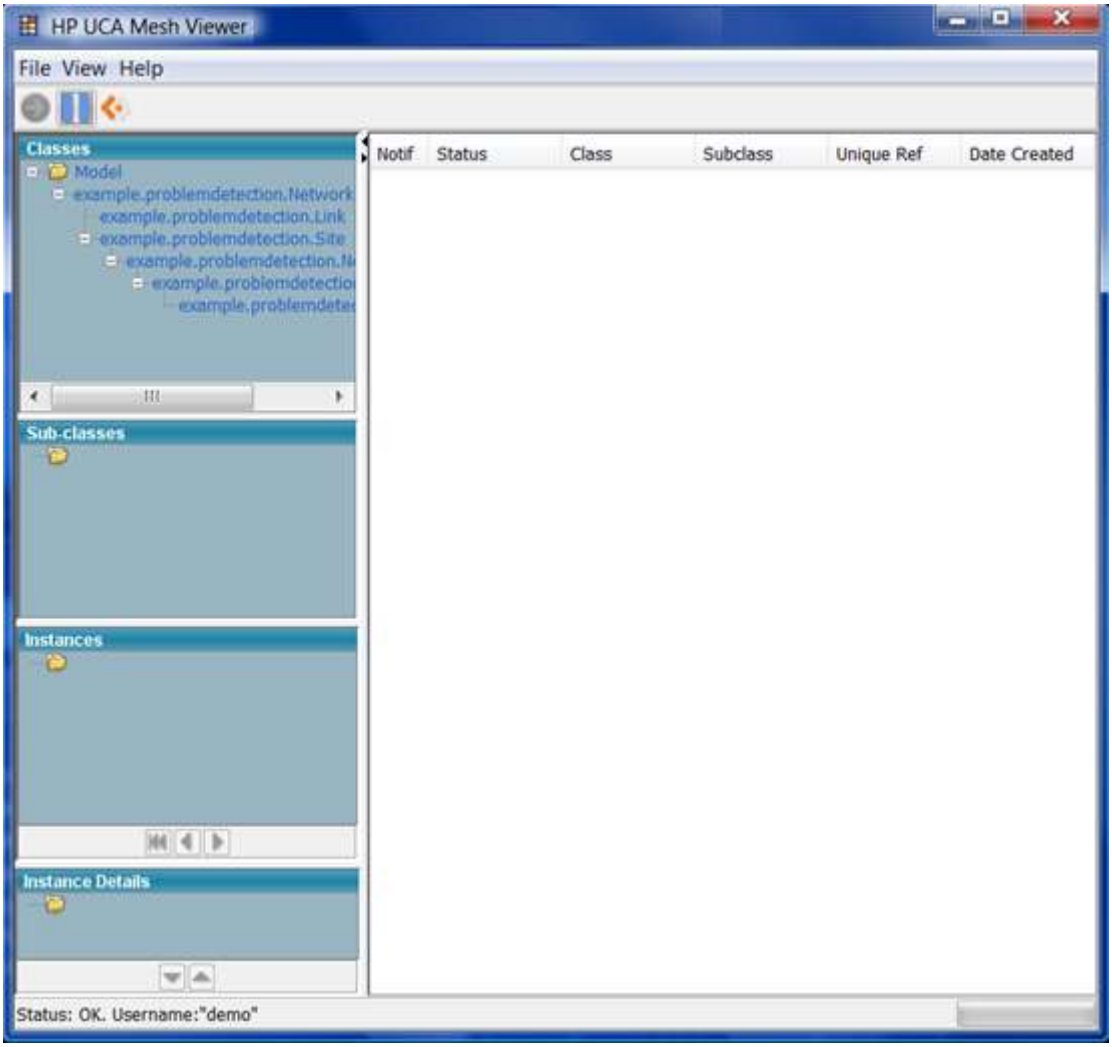

Figure 21: UCA's Mesh Viewer window

In the left pane, you can browse through the currently loaded objects.
The right part is used to display the object with a failed or degraded state.

9.2.6 Check deployed rules

From the UCA main page, press now the "Scenario Manager" button to launch the Scenario Manager applet. It is the development user interface to edit, load or see rules.

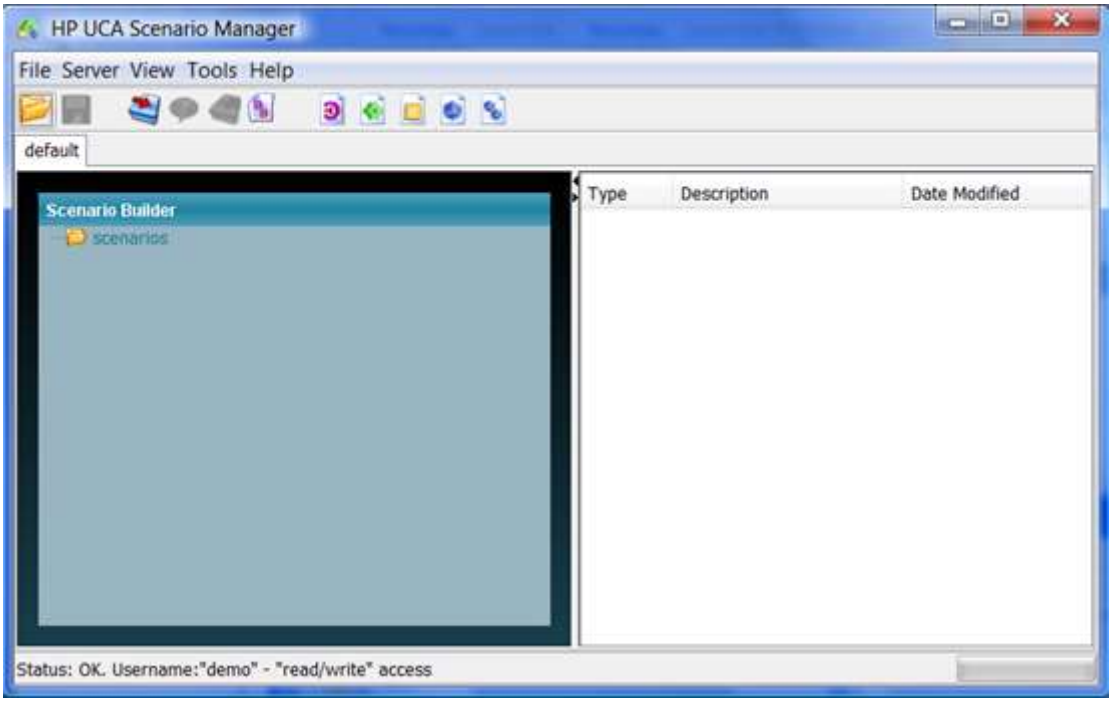

Figure 22: UCA's Scenario Manager window

To check the deployed rules, click on the “Load Current Deployment From Server” button (red arrow icon).
Then click on the example.problemdetection Tab.

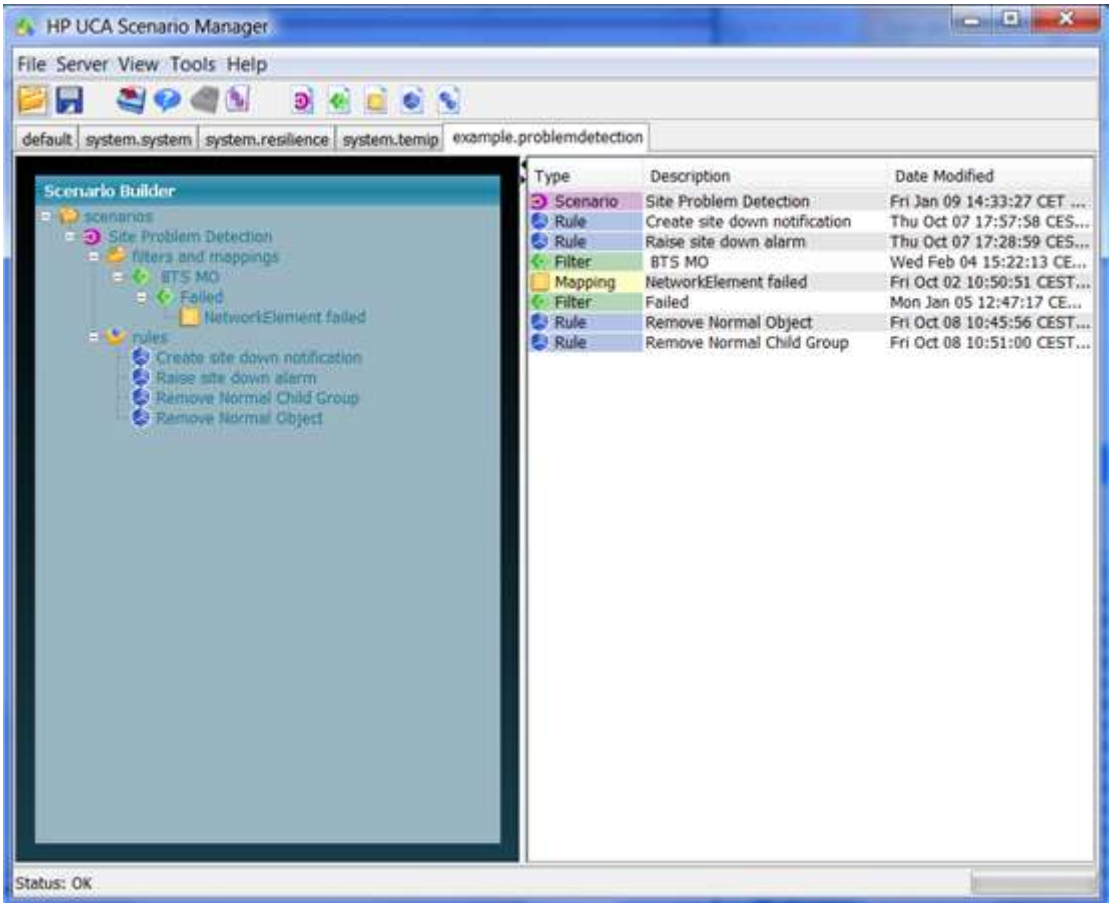

Figure 23: Scenario manager with the ProblemDetection rules loaded

You can see that two scenarios are deployed in the server:
Resilience Failover and recovery – which a system scenario used for internal processes monitoring
Site Problem Detection – which is our current example.

The ProblemDetection example configuration is now completed on the UCA side.

We will now configure the TeMIP side in the next sections.

9.2.7 Load the test MSL

The ProblemDetection will create new a TeMIP alarm on the Site object, and BTS alarms will be created. We therefore need these new classes in the TeMIP dictionary.
To do see, execute the following commands from the temip or root account.

```
# cd msl
# load_msl.sh
```

You can eventually check with the TeMIP dictionary browser (mcc_dap_browser) that the new Site, BSS and BTS classes are present in the dictionary.

9.2.8 Create the demo Operation Context

Create the Operation Context in TeP

```
# cd fcl
# manage do create_oc.cmd
```

This command creates a new Operation Context named “oc”.

9.2.9 Start the TeMIP-UCA integration processes

Please refer to the TeMIP Collector and TeMIP Remote Handler sections to see how to configure these two components that make the link between TeMIP and UCA. (You may already have done this during the UCA setup phase).

Usually, they are started automatically after the uca_start command.

Check with the uca_show command:

```
# su - uca
# uca_show
```

In case your server is configured in “Standalone” mode (not the default) the processes are not started automatically. In this case only, here is how to start them.

The TeMIP Remote Handler must be started before the Collector so that when the first alarms come in, and rules triggered by UCA, the Remote Handler is ready to execute output actions to TeMIP.

To start the TeMIP Remote Handler manually, use the following start-up script:

```
# su - uca
# $UCA_HOME/UCAremoteHandler/bin/runRemoteHandler.sh
```

Edit the `$UCA_HOME/UCAcollector/configuration/TeMIP_configuration.xml` file for the TeMIP collector to add the Operation Context “oc” in the collector operation context list, as follows:

```
<OperationContexts>
      <OperationContext>oc</OperationContext>
</OperationContexts>
```

Once configured (mainly with the correct hostname and OC to monitor), start the Collector with the following script, being the uca user:

```
# su - uca
# $UCA_HOME/UCAcollector/bin/runCollector.sh
```

9.2.10 Simulate events

At this stage, everything should be in place to be able to run the demonstration example.

There are fcl scripts to let you send the events and trigger the rules.

Emit the first BTS alarm, and eventually check the Mesh Viewer that the bts-1 object is failed.

```
# cd fcl
# manage do send_bts1_down.cmd
```

Then, send the 2 remaining BTS alarms:

```
# manage do send_bts2_down.cmd
# manage do send_bts3_down.cmd
```

Here, the rules should have fired and should see the Site object with a failed state in the Mesh Viewer.

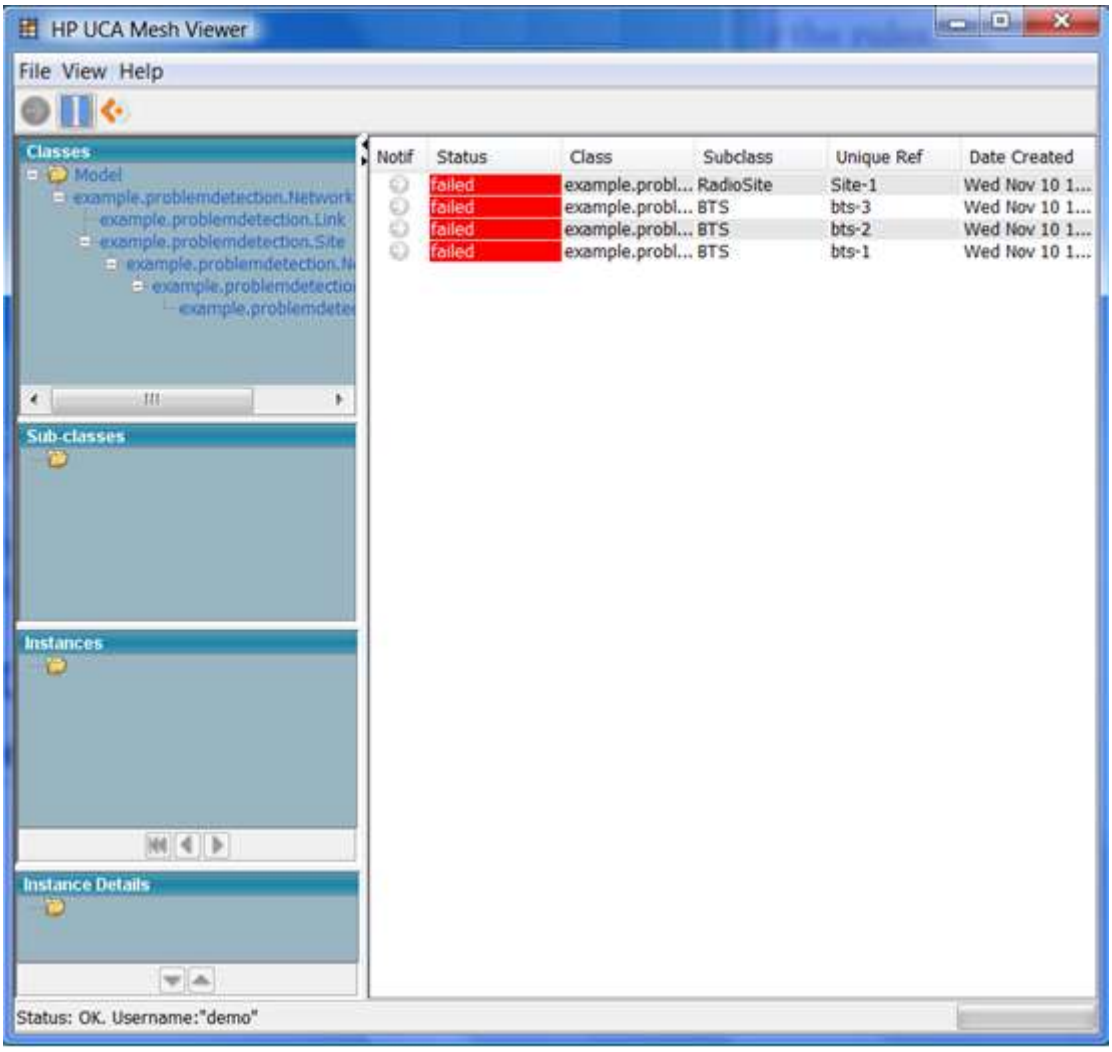

Figure 24: Mesh Viewer after the BTS alarms reception

You can also launch the “Fired Rules Viewer” from the System Manager “Tools tab”. You will then see the 2 rules that have fired for this scenario:

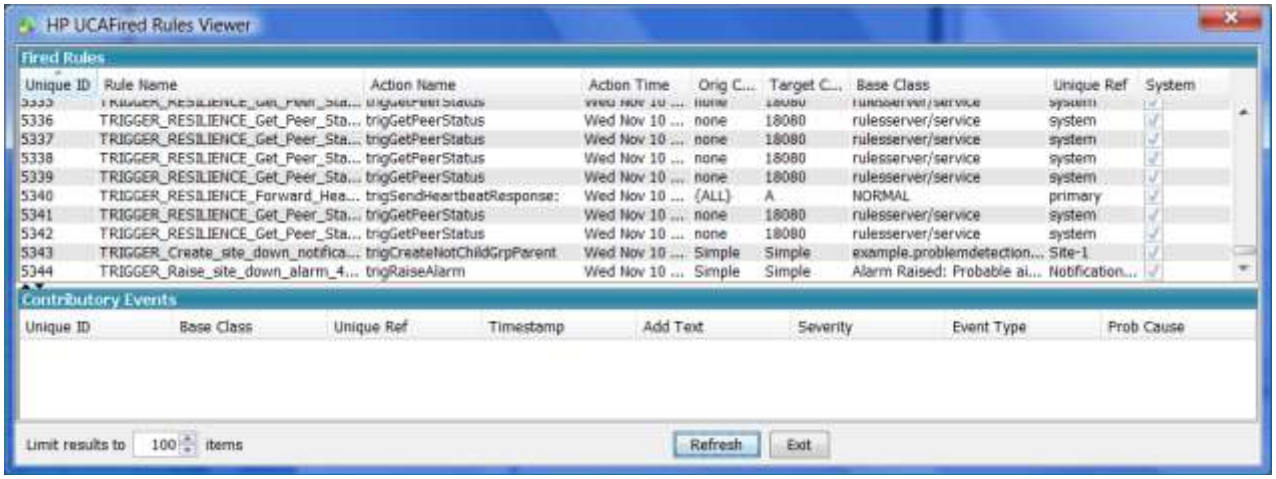

Figure 25: Fired Rules Viewer after the BTS alarms reception

And finally, more importantly, in the TeMIP Client you should see the new SITE alarm created by UCA, with the Correlation Tag attribute equal to "PROBLEMREPORT".
It means that our scenario has achieved its problem detection target. The 3 BTS down alarms have been replaced by a unique SITE problem alarm grouping the 3 others.

Figure 26: new SITE alarm created in TeMIP

9.2.11 Navigate through correlated alarms

If you double-click on the SITE alarm, you can navigate to the contributory alarms of the problem, which are the BTS alarms.

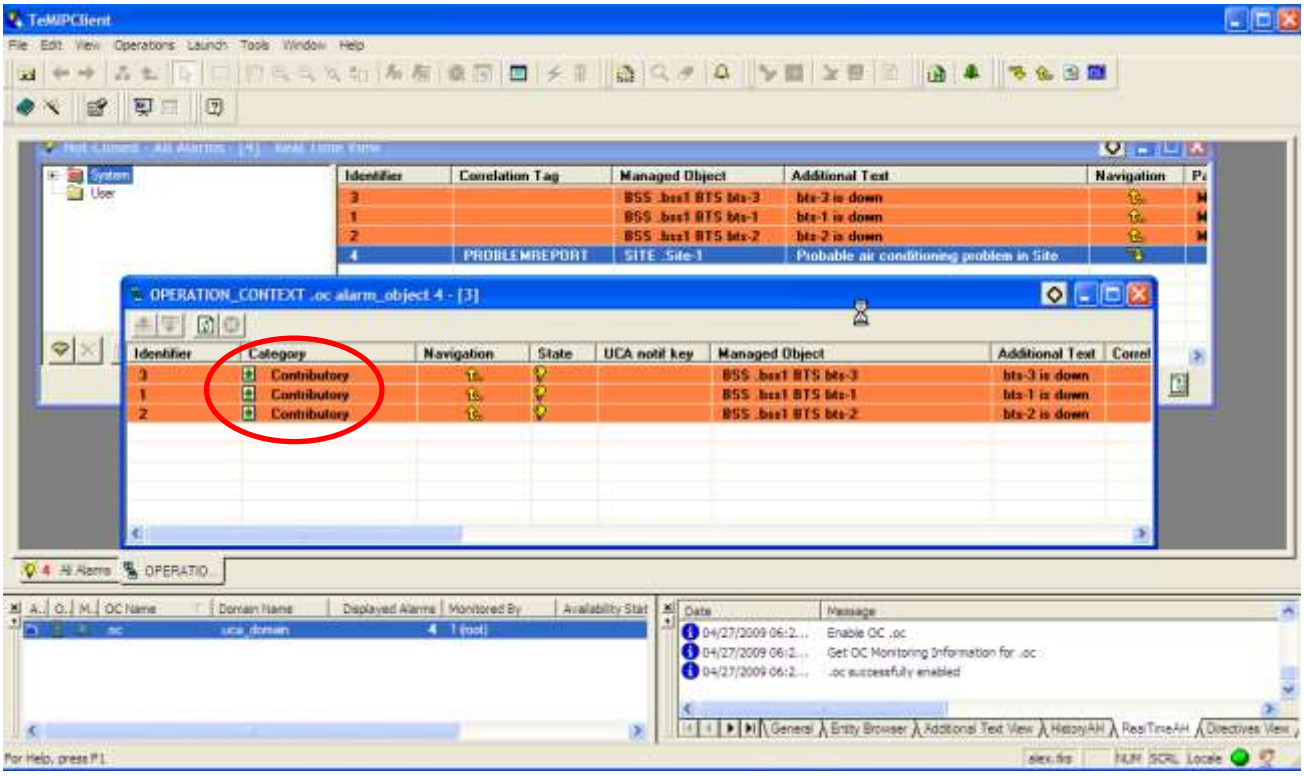

Figure 27: Alarm navigation example

Notice the "Category" column added to the tabular view.
You can also go back to the "parent" problem alarm by using the navigation buttons (yellow arrows).

Chapter 10 Service impact and RCA example

This is a more elaborated scenario, based on a service impact phase on a UMTS service, followed by a "root cause analysis" phase to retrieve all alarms participating to this service degradation.

Of course, one can use the step by step description detailed for the previous example to run this scenario. The steps are exactly the same, and the example directory is structured in the same way.

10.1Model

The model is somehow the following one (you can see the full UML one in the model directory):

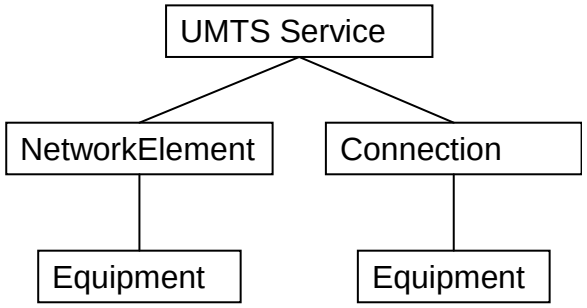

Figure 28: Service Impact example, Meta Model

The model is fairly generic and can be reused for many other network representations, even though the class names does match very well with the reality. For instance, in our example, the "Cell" objects of the UMTS network become instances of the "Equipment" class, what is not very logical.
This is a typical trade-off to do: re-use an existing simple and generic model and use the UCA sub classing concept (please refer to the UCA user's guide for details), or write a new specialized model each time, which strictly match the network topology.
Working on a specialized model can ease the rules writing, since the domain specific logic (e.g SDH) is easier to express, whereas using a generic model enables re-usability, especially rules templates and patterns.

In the example, the model is instantiated with the following mesh objects populated, with their relationships.

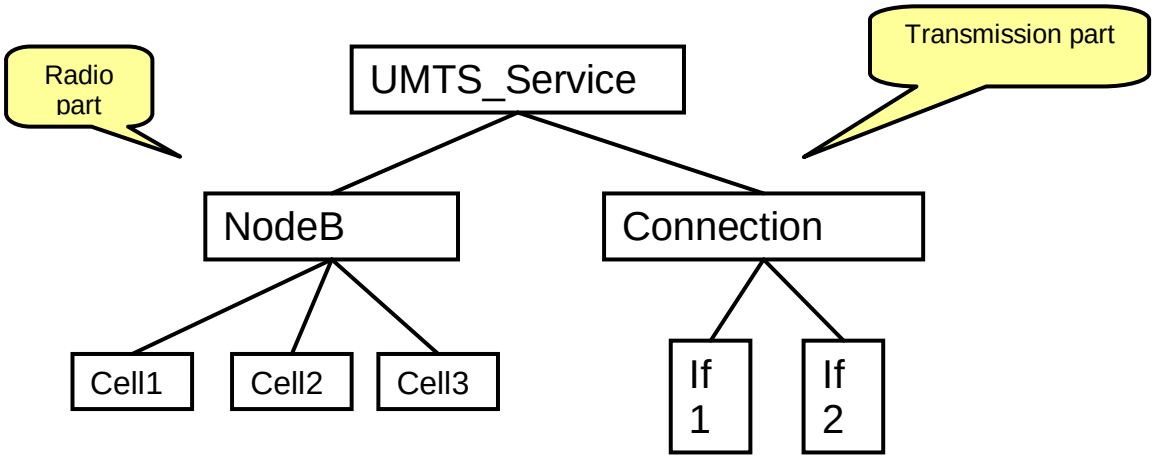

Figure 29: Service Impact model, Instantiation

The idea of the scenario is to have a service relying on two separate part of the infrastructure: a radio part and a transmission part. The service can be affected if either one or the other part is down. In each sub-system, we implement problem detection rules to detect either a radio problem (one or several cells down) or a transmission problem (two ends of a connection down). If a problem is detected, it is propagated up to the service, to generate a new Service Impact alarm in TeMIP.

10.2 Transmission problem detection

A Connection is modeled as an "Associate Group" with two ends, which represent network interfaces. The connection is detected as down when its two end interfaces are down.

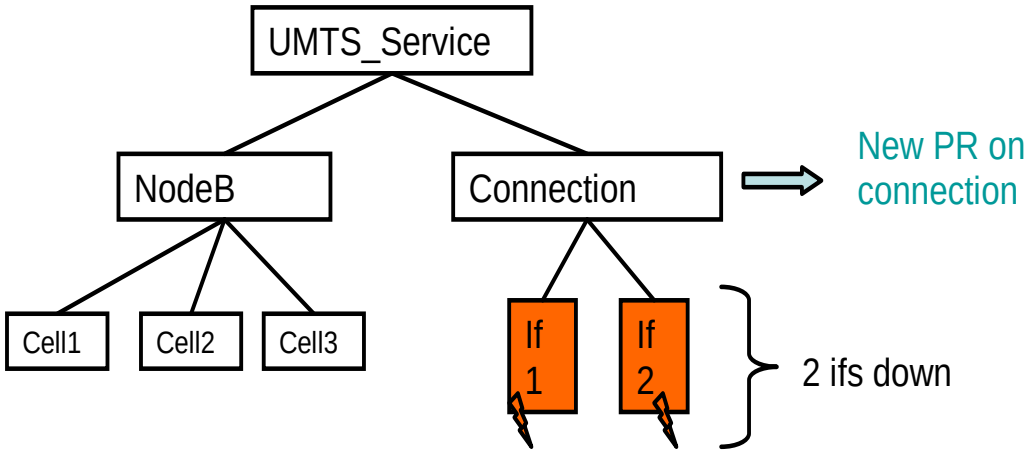

Figure 30: Transmission problem detection

In TeMIP, a new Problem Report alarm is created on the Connection managed Object. This Problem Report alarm associates the two initial Interface Down alarms, marked as "Contributory" to the problem.

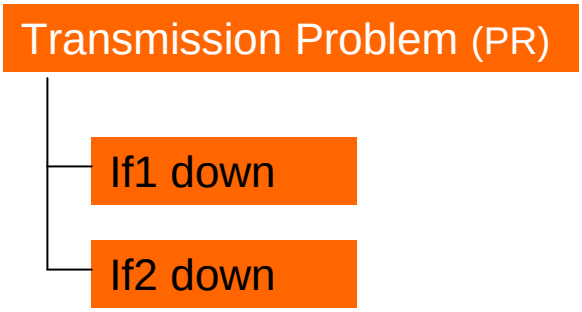

Figure 31: New transmission alarm in TeMIP

10.3 Radio problem detection and Service Impact

On the radio part, we also have a problem detection pattern in place. It is trigged when we receive a "cell down" alarm. A new Problem Report alarm is then created in the NodeB managed object. The state of the NodeB is then propagated to the above UMTS service. Because this one is now degraded a new Service Impact alarm is therefore created in turn into TeMIP.

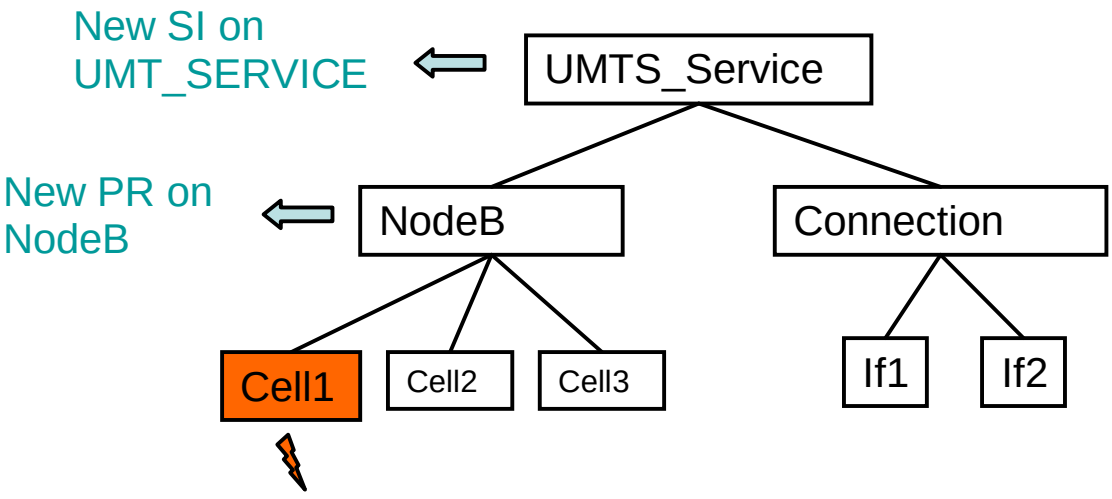

Figure 32: Radio problem detection, and service impact up to UMT service

In the end, we have the following linkage between alarms in TeMIP. The operator sees only one alarm.

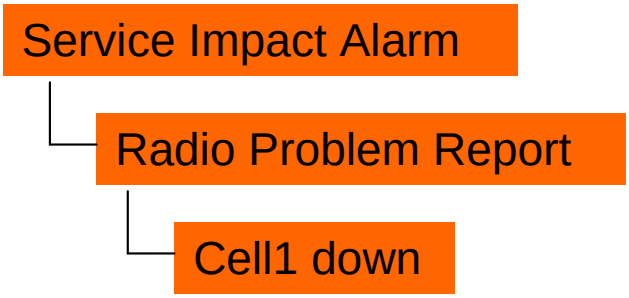

Figure 33: New Radio Problem and Service Impact alarms in TeMIP

10.4 Severity increase

Now, if all cells related to a NodeB are down, we wish to increase the severity of the Radio Problem and UMTS Service Impact alarms to critical.

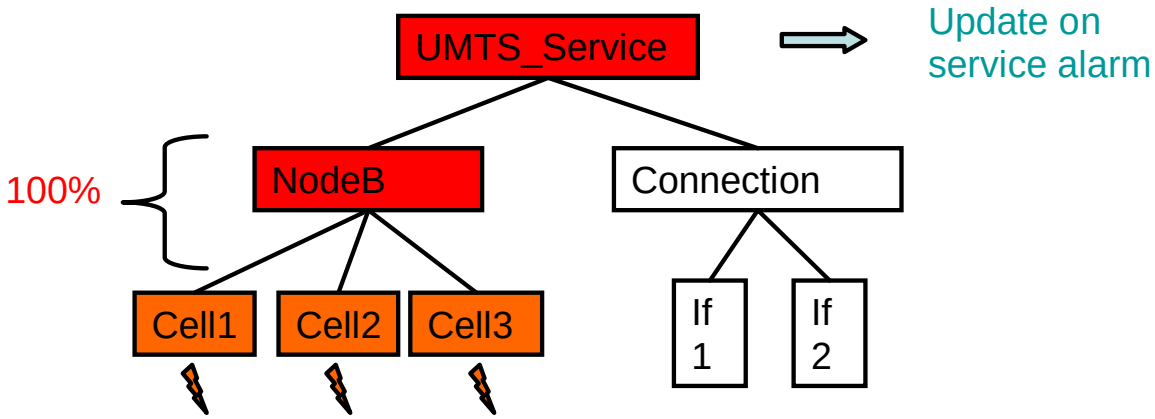

Figure 34: Severity escalation on Service

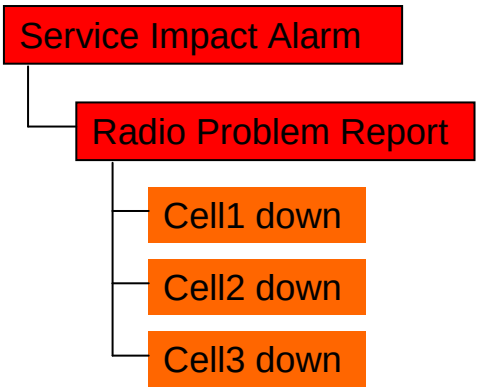

Figure 35: Severity escalation in TeMIP

10.5 Final picture

Finally, as a “cross domain” example, we can see that both the Radio and Transmission problem participate to the same service degradation.

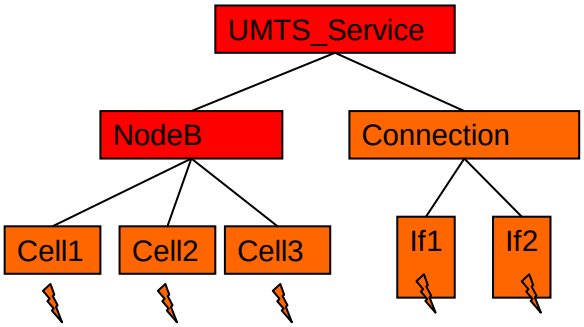

Figure 36: Service impact scenario, final picture

Thanks to UCA, this is now visible in TeMIP with the following alarm hierarchy.

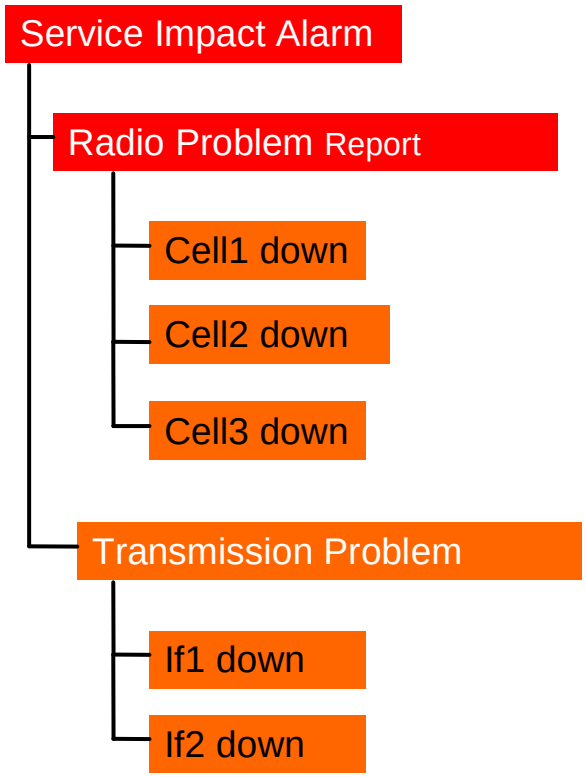

Figure 37: Service impact scenario, alarms correlated in TeMIP

In the end in TeMIP, the operator sees only one alarm, the one on the service, from which he can drill-down to the associated alarms. In the picture above, the alarms with a white font are the ones created by UCA, whereas the ones with a black font are the 4 ones received initially in TeMIP and "demoted" under the new created ones (master).

Glossary

This glossary contains definitions of terminology used in the TeMIP User Documentation set.

Agent

The portion of an entity that performs management procedures on behalf of a director, receiving requests from, and returning responses to, the director. TeMIP supplies off-the-shelf Agent functionality for OSI networks through a dedicated Presentation Module, the OSI PM.

Alarm

A condition or occurrence in a managed network that is recognized as requiring notification to a user for further analysis, possibly leading to corrective action.

Alarm Objects

Alarm Objects are entities derived from alarms generated by network elements, which can be handled and manipulated using AH NT. Alarms that satisfy the Alarm Handling filtering criteria are transformed into Alarm Objects.

Attribute

A piece of information that describes an entity such as a status or a characteristic. A property of an alarm object. An attribute has a value.

Alarm Rule

A user-defined logic statement that specifies an alarm condition to be detected and passed to the Notification FM.

Dictionary

The dictionary is a shared information store available to all management modules. It is replicated on each director.

The dictionary contains the definitions of all global classes, including their child classes, their attributes, their events, and the directives that they support.

Director

A software system that interacts with a user, initiates management operations on behalf of the user, coordinates management activities with entities, and provides high-level management applications.

Discriminator

An OSI-compliant data structure that filters the received event reports, allowing only those that satisfy the specified criteria to be passed through.

Entity Model

An entity is an item in a model stored in a database, representing a real-world object or concept. The TeMIP Entity Model exists for the purpose of network management. It provides a framework for extensible architectures for managed objects.

The only network management actions currently initiated by an entity as opposed to by a director, are the processing of events into event reports and the forwarding of event reports.

Entity Hierarchy

A set of entities defined in the TeMIP management model comprising one ancestor entity and all its descendants.

Event

An occurrence of a normal or abnormal condition detected by a network element that might be of interest for network management.

Filters

In an Alarm Handling context, filters allow for the specification of criteria that alarm objects must meet in order to have a handling function performed. Filter patterns are used to determine whether or not an alarm object should appear in the alarm list. The filter pattern is expressed in terms of the presence or value of certain attributes of the alarm object, and is satisfied if it evaluates to TRUE.

Framework Command Line (FCL)

A user interface comprising a command line and command language, which essentially duplicates the services of the iconic map but without its graphical representations. The FCL commands are used to apply management functions to managed objects. They are specifically useful when there is a requirement to manage a network from a non-graphical terminal.

Managed Object

A network element that is managed.

Operation Context

An independent and self-contained view of a management domain that defines an instance of alarm handling to achieve a specific management objective.

OSS-J

Operational Support System through Java. It defines and standardizes a set of XML and Java APIs that facilitate the integration of OSS products with each other and makes it almost seamless.

TeMIP Framework

Digital object-oriented management product (framework and applications).

TeMIP Operator or User

The owner (in the OS sense) of an application process invocation.

TTR

Trouble Ticketing Report. Raised against one or more alarm reports to initiate repair actions.