hp Unified Correlation Analyzer

Unified Correlation Analyzer
for
Event Based Correlation

Version 3.3

User Interface Guide

Edition: 1.0

September 2015

Legal Notices

Warranty

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

License Requirement and U.S. Government Legend

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

Copyright Notices

Trademark Notices

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

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

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

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

Firefox® is a registered trademark of the Mozilla Foundation.

Google Chrome® is a trademark of Google Inc.

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

UNIX® is a registered trademark of The Open Group.

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.

Contents

Figures

Tables

Preface

This guide describes how to use the web-based administration user interface of UCA for EBC (Unified Correlation Analyzer for Event Based Correlation).

Product Name: Unified Correlation Analyzer for Event Based Correlation

Product Version: 3.3

Kit Version: V3.3

Intended Audience

Here are some recommendations based on possible reader profiles:

- Solution Developers
- Software Development Engineers
- Solution administrator
- Solution operators

Software Versions

This guide applies to all supported platforms (Linux, HP-UX, and Windows).

As the provided user interface is web based, the rendering of some components may be slightly different depending on the browser used.

However the described functionalities should be identical on any supported browser.

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

The following documents contain useful reference information:

References

[R1] *UCA for Event Based Correlation - Reference Guide*

[R2] *UCA for Event Based Correlation - Value Pack Development Guide*

[R3] *UCA for Event Based Correlation - Administration, Configuration and Troubleshooting Guide*

Support

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

The Software support area of the Software Web site includes the following:

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

Chapter 1

Introduction

This guide describes the web-based administration user interface of UCA for EBC which provides the following functionalities to the UCA for EBC product:

- Monitoring,
- Administrating,
- Troubleshooting.

Note

It is strongly recommended to read the "*HP UCA for Event Based Correlation - Administration, Configuration and Troubleshooting Guide [R3]*" for a better understanding of this document and the different user interface snapshots.

1.1 Software pre-requisites

1.1.1 Supported web browsers

The UCA for EBC User Interface is web-based and thus is accessible through a web browser.

The list of supported browsers is the following:

Browser	Version
Microsoft Internet Explorer	**8 (or later) see Note**
Mozilla Firefox	**36.0 (or later)**
Google Chrome	**41.0 (or later)**

Table 1 - Supported web browsers

Note to Internet Explorer users

The UCA for EBC User Interface works better with Internet Explorer 8.0.

Internet Explorer 9, 10 and 11 must be set with the "Browser Mode" set to "Internet Explorer 8 Compatibility view" mode.

Internet Explorer compatibility mode can be set from the following menu:

Tools -> F12 Developer Tools -> Browser Mode

1.2 Launching the UCA for EBC User Interface

Ensure that the UCA for EBC Server is started before launching the user interface. If UCA for EBC is not started, you will get an error from your web browser indicating that it cannot connect to the web server.

When trying to connect to the UCA for EBC user interface, it is best to use the fully qualified DNS name of the system running UCA for EBC Server.

If UCA for EBC Server is running on your local host, you can use “localhost” as the name of the host to connect to using your web browser.

http://localhost:8888/uca/

Otherwise, the UCA for EBC User interface is accessible at the following URL:

http://*<hostname or IP address>*:*<port #>*/uca

<hostname or IP address> should be replaced by the actual hostname (full DNS name) or IP address of the UCA for EBC Server system.

<port #> is the port number for UCA for EBC User Interface, 8888 by default.

This port number can be changed thanks to the **uca.gui.port** property in the uca-ebc.properties file.

1.3 UCA for EBC User Interface layout

The following picture shows the UCA for EBC User Interface Main screen (also called ‘dashboard’).

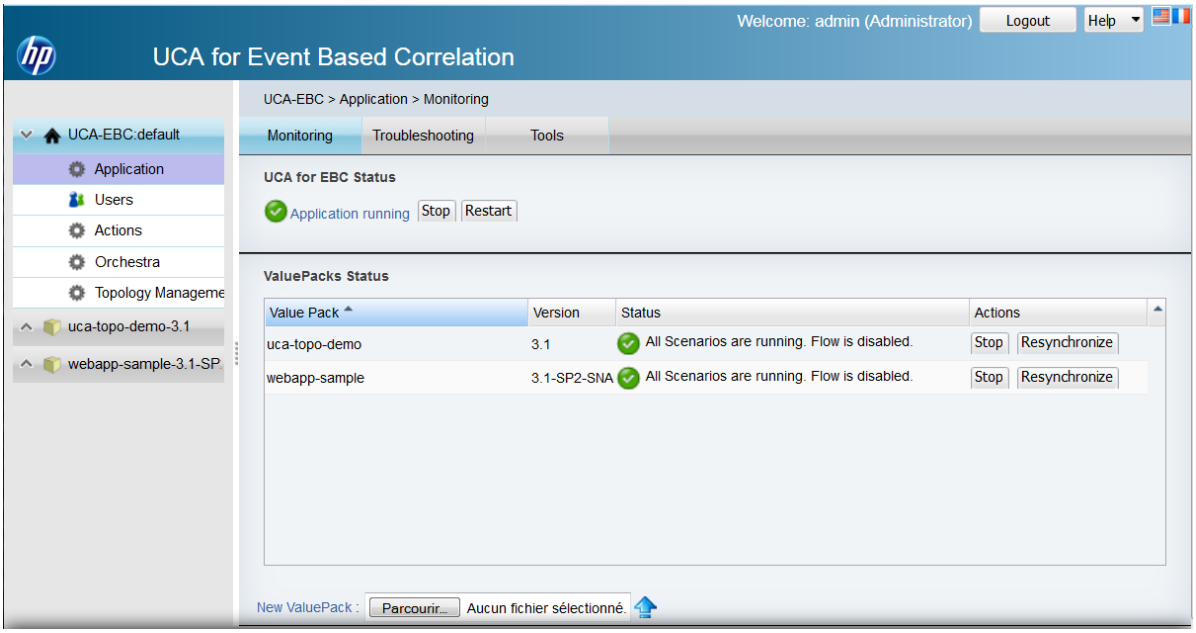


Figure 1- UCA for EBC Application Monitoring View (or Dashboard)

The UCA for EBC User Interface screen can be split into five separate sections that are read from top to bottom and from left to right:

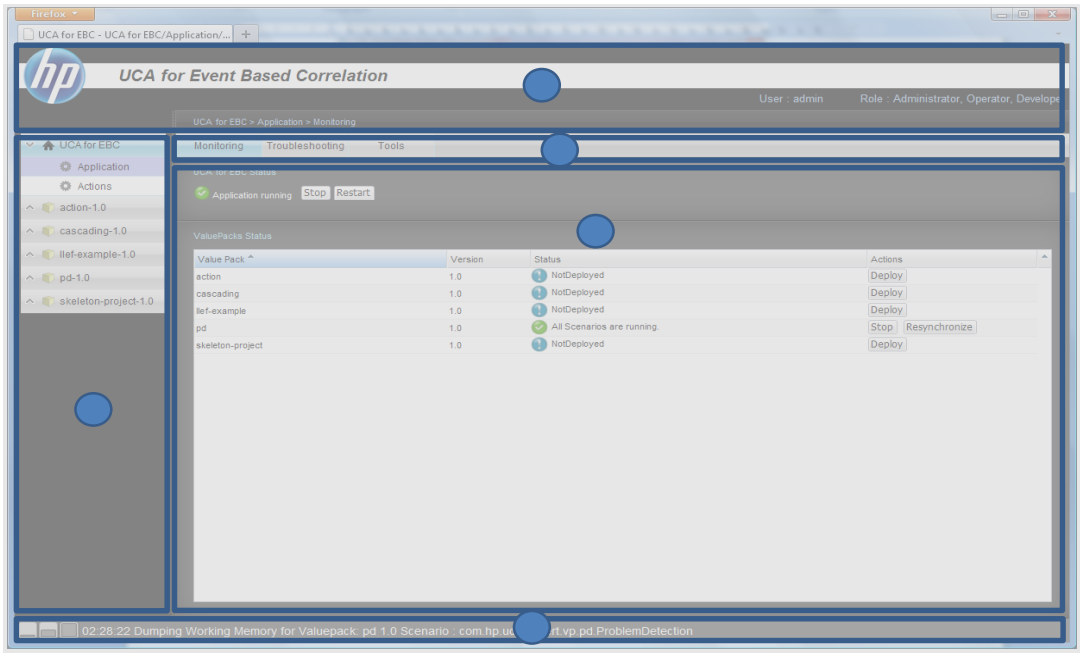


Figure 2 - UCA for EBC User Interface layout explained

Below is a table that lists the different sections in the UCA for EBC User Interface:

Area	Explanation
Section 1	Header section
Section 2	Main menu section
Section 3	Sub-menu section
Section 4	Content area section
Section 5	Console section

Table 2 - UCA for EBC User Interface layout explained

The following paragraphs contain additional details on each section of the UCA for EBC User Interface.

1.3.1 Section 1: Header section

Section 1 is a header section that gives information about the application name, the logged-in user and the user roles associated with the logged-in user.

The bottom of this header section displays a breadcrumb trail that helps navigating the UCA for EBC User Interface by providing information on what is currently being displayed in the content area of the GUI (in Section 4).

What is displayed in the content area is the result of what the user has selected from both the left-hand side main menu (in Section 2) and the top horizontal sub-menu (in Section 3).

This section holds a Help button that allows user to:

- Have access to the Java documentation brought by the UCA for EBC product

- Check if some other instances are running on the same server and possibly access to the other UCA for EBC GUIs
- Have the current version of the UI (build number and date of build)

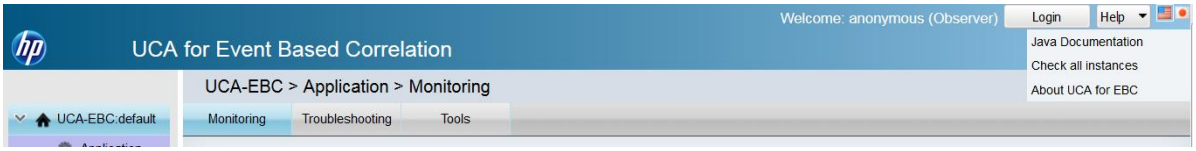


Figure 3 – Header section

This section also holds some country flags in order to have direct access to the internationalized version of the UI: defaults are US and Japan. For your specific flag, ask the UCA for EBC product management or support team.

1.3.2 Section 2: Main menu section

This is the main menu of the UCA for EBC User Interface. This menu is implemented as a "Stack Menu". This means that by clicking on one of the header menu entries, the menu will display the underlying sub-menus of the selected entry.

When the sub-menu list is displayed, at least one of the sub-menus is selected (by default the first sub-menu on the list is selected). A simple click on one of the sub-menu items will select this sub-menu.

The first header menu entry is named "UCA for EBC" and represents the application itself. At GUI start-up this entry is selected by default. The other header menu entries represent the Value Packs installed on the UCA for EBC Server. This means that this list will differ from one installation to another, depending on the Value Packs installed on UCA for EBC Server.

1.3.3 Section 3: Sub-menu section

This horizontal menu is dependent on the selection made on the left-hand side header menu (section 2).

For example, if the header menu selection is **"UCA-EBC:default -> Application",** the sub-menu section will display sub-menus specific to this selection. If the header menu selection is different (for example: one of the Value Packs is selected) then another horizontal sub-menu will be displayed, and this sub-menu will be specific to this new selection.

1.3.4 Section 4: Content area section

The content area section is most important of the UCA for EBC User Interface. It is where the application information data is displayed. The data displayed is contextual and depends on:

The item and sub-menu item selected on the left-hand side main menu

The item selected on the top horizontal sub-menu

At application startup the default selection is the following:

Left-hand side main menu: "UCA-EBC:instanceName -> Application"

Top horizontal sub-menu: “**Monitoring**”

Also, at application startup, the breadcrumb trail should display:

UCA-EBC:instanceName > Application > Monitoring

Note that the “**default**” is the instance name of UCA for EBC (this could differ depending on your configuration).

At application startup, the “UCA for EBC application dashboard” is displayed.

1.3.5 Section 5: Console section

Section 5 is the UCA for EBC User Interface console. The console logs important events: connection problems with the server, administrative actions acknowledgement and unexpected exceptions. By default, only the last logged line is visible. Some buttons on the left side of the console allow you to expand the console window to half-screen or full screen size so that you can view more than the whole content of the console.

Chapter 2

UI Authentication

Up to version 3.2 of UCA for EBC, no special authentication is required to access the UI page of UCA for EBC. An anonymous user was used with the role of “Observer”. Then the login procedure was part of the UCA for EBC home page if another role was required for accessing the UI.

From 3.3 onwards, for security reasons, a new mechanism has been put in place:

- All pages of the UI are now restricted to an authenticated user.
- User has to login before accessing the UI welcome page.
- No more login button is available once user is logged.

To log in UCA for EBC UI, user has multiple possibilities, and all may be tried seamlessly in sequence order:

- The legacy ‘users’ database that is managed locally by UCA for EBC Server instance
- A simple clear user/password/role user service (with values not encrypted)
- An access to a LDAP server

All above possibilities are allowed (or not) by configuring a new configuration file: `$UCA_EBC_INSTANCE/conf/security-context.xml`.

This file is relying on spring-security public schema to validate the elements defined within it. It declares mainly an authentication manager with its authentication providers listed.

Notes

- The spring-security schema: http://www.springframework.org/schema/security/spring-security.xsd
- The spring security namespace configuration: http://docs.spring.io/spring-security/site/docs/3.1.x/reference/ns-config.html

Let’s describe those possibilities in detail…

2.1 The legacy authentication provider: the ‘users’ database

This provider is the one used by previous versions of UCA for EBC. It is a simple database managed locally on UCA for EBC Server, handling usernames, passwords and roles in a single table.

The usernames and associated roles are stored in clear text and the passwords are encrypted.

To declare that provider in security-context.xml, there is nothing to do, as this is by default enabled.

To get rid of that provider, comment out the line:

```
<authentication-provider ref="usersDbAuthenticationProvider" />
```

For backward compatibility reasons, it is recommended to leave that provider in place for 3.3, so that when upgrading your UCA for EBC version, you do not lose your users configuration.

However, since this table is not designed to be used across multiple instances of UCA for EBC, it might be deprecated in future versions and replaced by a distributed database accessed through remote JDBC calls.

2.2 The simple in-memory user service provider

This provider is introduced in 3.3 to have a simple way to declare new user roles directly used to access UI.

As the anonymous login is no more authorized, there is a need to have the equivalent. Hence, a default user 'guest' with password 'guest' is declared in that provider with the role of 'Observer'.

For integration tests, it is easy to add users and roles directly in that provider.

However, for security reasons, in a production environment, it might be judicious to comment out totally the lines below in security-context.xml.

```
<authentication-provider>
    <user-service>
            <user name="guest" password="guest" authorities="Observer" />
    </user-service>
</authentication-provider>
```

2.3 The LDAP provider

This provider is introduced in 3.3 to have authentication based on a LDAP service. This provider is looking UCA for EBC UI roles directly in LDAP so that when an authenticated user has none defined, an 'Observer' role is given to him.

2.3.1 Defining LDAP server access

Before declaring an LDAP provider, you need to configure your LDAP server properties, with its URL, port number, root DIT, and bind DN and password.

This is done by declaring a spring bean, as per example below:

```
<ldap-server url="ldap://localhost:10389/o=SevenSeas" manager-
dn="uid=admin,ou=system" manager-password="secret" />
```

The above example is taken from a standard Apache Directory Server with its LDIF (o=SevenSeas) example loaded.

More specifically:

- **url** defines the protocol, the LDAP hostname, the port number and the root DIT
- **manager-dn** and manager-password define what will be used by default when binding to the server (instead of binding anonymously)

2.3.2 Defining LDAP provider

To declare an LDAP provider, you need to configure its bind DN and password (if not configured statically in LDAP server above), the search filter and base DN, but also how the roles (a.k.a. authorities) are loaded.

An example is given in security-context.xml as per below:

```
<ldap-authentication-provider user-search-filter="(uid={0})" user-search-base="ou=people" group-search-base="ou=groups" group-search-filter="uniqueMember={0}" role-prefix="none" />
```

More specifically:

- **user-search-filter** defines the search filter for looking up the provided username referred as {0}. If you want to search on an email attribute simply change to "(email={0})"
- **password** matching is done using the 'userPassword' attribute of the requested user
- **user-search-base** defines the base DN where to start searching
- **group-search-base** defines the base DN where to start searching for a role
- **group-search-filter** defines the filter for looking up the role of the provided username referred as {0}
- **role-prefix** should be set to "none"

2.3.3 Configuring user roles in LDAP

Only roles 'Administrator' and 'Developer' are useful to be defined in LDAP as the default role 'Observer' is given to any user authenticated in LDAP.

Given the example of group-search attributes above, defining roles could be done as:

1. Declare an entry 'Administrator' under 'uca' subtree of 'ou=groups'
2. Add an attribute 'uniqueMember' with full DN for each user who has this role

As per following LDIF, we can see that James Hook and Horatio Nelson are administrator, whereas all other people defined in 'ou=people' tree are given the role of 'Observer'.

```
dn: cn=Administrator,ou=uca,ou=groups,o=SevenSeas
objectclass: groupOfUniqueNames
objectclass: top
cn: Administrator
uniquemember: cn=James Hook,ou=people,o=sevenSeas
uniquemember: cn=Horatio Nelson,ou=people,o=sevenSeas
```

Notes

- The spring security LDAP authentication reference : http://docs.spring.io/spring-security/site/docs/3.1.x/reference/ldap.html

- If the LDAP Secured protocol is to be used, make sure to have defined properly the **javax.net.ssl** properties with your **trusted certificate** in `$UCA_EBC_INSTANCE/conf/uca-ebc.properties`.

Chapter 3

UCA for EBC Administration

The UCA for EBC User Interface allows an UCA-EBC user to perform administrative operations.

Each operation is accessible or not from the UI depending on the Role of the connected user. There are three different roles: **Observer**, **Developer**, and **Administrator**.

When the web interface is stated, no user is connected and the role is automatically set to Observer.

The following table lists the accessible operations depending on the user Role:

Level	Operation	Observer	Developer	Administrator
Application Level	Application Monitoring (Dashboard)	✓	✓	✓
	Stop/Restart the application			✓
	Manage Users			✓
	Topology data load (if feature installed)		✓	✓
	Application tooling			✓
	• reload trace configuration			✓
	• clean up log Database	✓	✓	✓
	Application troubleshooting	✓	✓	✓
	• statistics	✓	✓	✓
	• logs	✓	✓	✓
	Actions	✓	✓	✓
	• troubleshooting	✓	✓	✓
	• display configuration (*)		✓	✓
	• modify configuration (*)			
Value Pack Level	Value pack monitoring	✓	✓	✓
	• scenarios list	✓	✓	✓
	• mediation flows list (*)	✓	✓	✓
	Deploy/Undeploy a value pack		✓	✓
	Start/Stop a value pack		✓	✓
	Start/Stop a mediation flow (*)		✓	✓
	Resynchronize a mediation flow (*)		✓	✓
	Display mediation flows configuration (*)	✓	✓	✓
			✓	✓
	Modify mediation flows configuration (*)		✓	✓

Level	Operation	Observer	Developer	Administrator
	Save as new value pack (*)	✓	✓	✓
	Value Pack troubleshooting	✓	✓	✓
	• statistics	✓	✓	✓
	• logs			
Scenario Level	Scenario Monitoring	✓	✓	✓
	• rules list	✓	✓	✓
	• rules files list (*)	✓	✓	✓
	Dump the working memory of a scenario		✓	✓
	Clear the working memory of a scenario		✓	✓
	Reload a scenario		✓	✓
	Reset the status of a scenario		✓	✓
	Remove a rule (*)		✓	✓
	Load/Reload/Unload a rules file (*)	✓	✓	✓
	Display scenario configuration		✓	✓
	Modify scenario configuration (*)	✓	✓	✓
	Scenario troubleshooting	✓	✓	✓
	• statistics	✓	✓	✓
	• logs			

Table 3 - UCA for EBC User Interface operations by level

3.1 Users logging and roles

3.1.1 User roles

UCA-EBC provides three different roles:

Observer: The Observer role is a read-only role. Only monitoring of the application is available, no particular operation is allowed.

Developer: The Developer role allows a value pack developer to manage value packs. All operations on value packs and scenario are available (deploy / start /stop etc…)
Some useful operations on the application are also available such as reloading the trace configuration file, or launching a topology data load (if this feature is installed on the system)

Administration: The Administration role gives the user the full access to all operations on the UCA-EBC system includes the full stop / re-start of the application from the GUI.

3.1.2 User logging

When the UCA-EBC web interface is started no user is logged-in.

Since UCA-EBC 3.3, anonymous user is no more authorized to get access to the UI, so any user should go through the login page before accessing the UI.

At UCA-EBC installation:

- A guest user is configured within the in-memory users service provider
- An administrator user is created in the 'users' database provider

Above users are defined with following default credentials:

User Name	**Password**	**Role**
guest	guest	Observer
admin	admin	Administrator

Log as the 'admin' user to create additional users in the 'users' database provider

3.1.3 User Management

This section applies only to the 'users' database authentication provider.

Configuring users in LDAP for the LDAP provider is out of scope of this document.

Adding, removing or changing users can only be done by a user with Administrator role.

The user Management Panel is reached by selecting the **UCA-EBC:instanceName > Users > Configuration**

This lead to the following panel:

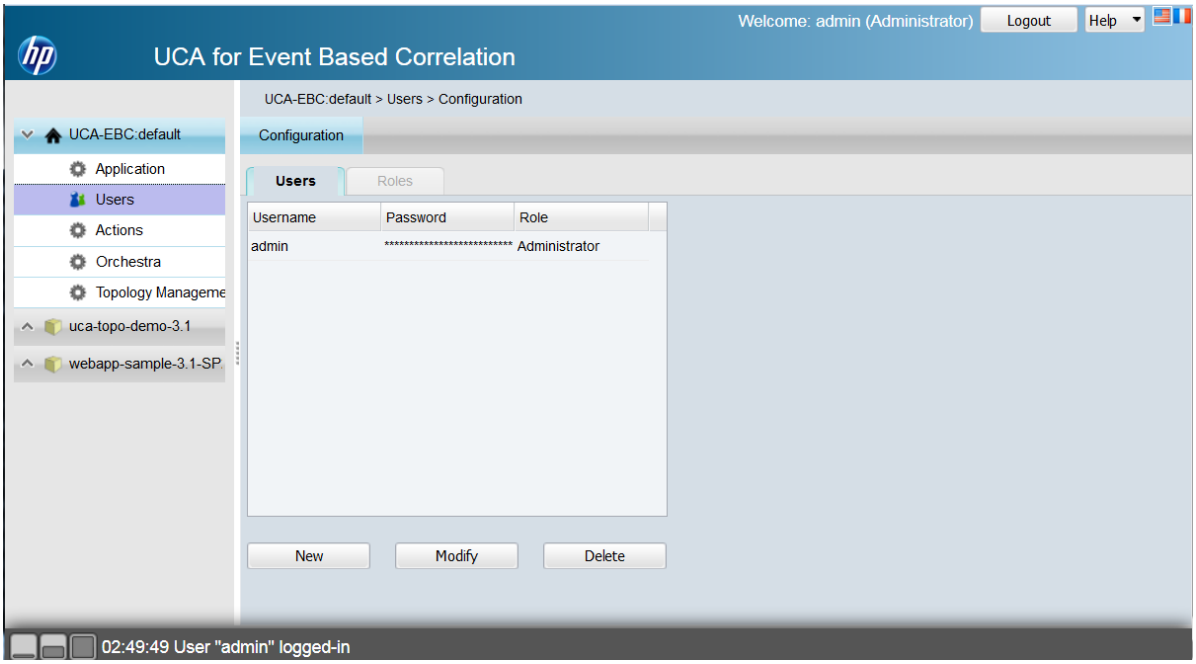


Figure 4 - User Management Panel

Use the New, Modify or Delete buttons to add change or remove users.

3.2 UCA for EBC operations

3.2.1 Application level operations

Application level operations are accessible from the dashboard window, which can be accessed by selecting the **UCA-EBC:instanceName > Application > Monitoring** menu as shown in the screen capture below:

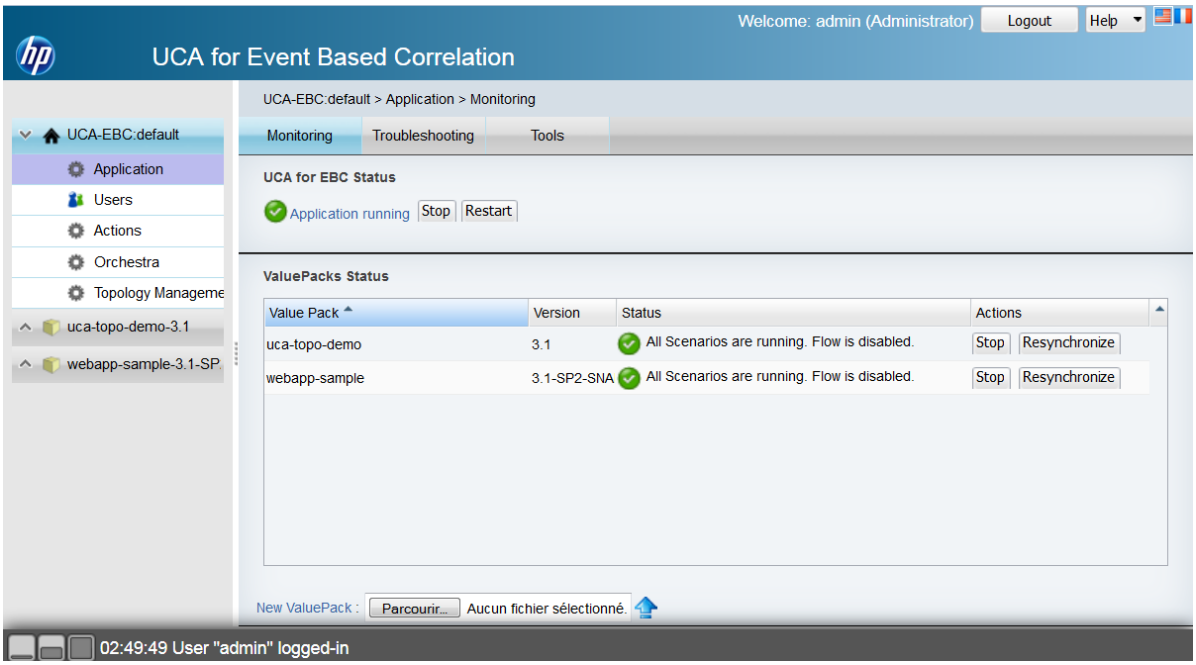


Figure 5 - UCA for EBC Application level operations

From the UCA for EBC Status section, the Stop button allows you to stop the UCA for EBC Server running this user interface and the Restart button allows you to stop and restart the UCA for EBC server.

Note

When the UCA for EBC Server is stopped instead of being restarted, the embedded web server running the UCA for EBC User Interface is also stopped.

This means that the UCA for EBC User Interface will not be able to connect to the server anymore and will become unavailable until the UCA for EBC Server is manually restarted directly on the system running UCA for EBC Server.

3.2.2 Value Pack level operations

The Value Pack level operations are accessible either from the Application Monitoring View (or Dashboard) or from the Value Pack Monitoring View: **valuepack_name > ValuePack > Monitoring**.

The operations available on any given Value Pack are dependent on the state of this Value Pack.

The following picture explains the Value pack life cycle within the UCA for EBC product:

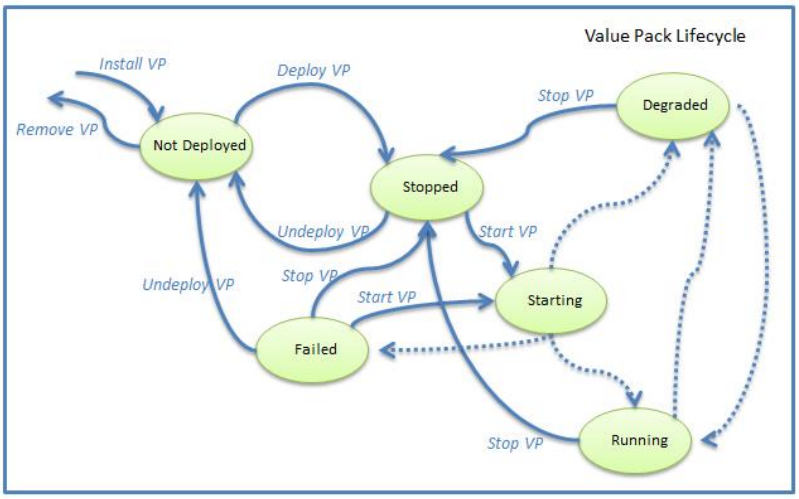


Figure 6 - Value Pack life cycle

Bold lines transitions indicate a specific action on a Value Pack (deploy, start, stop, etc...).

Dotted lines transitions indicate either internal processing or a problem:

Running state: all scenarios are in "Running" state and so is the mediation.

Failed state: in case of XML file configuration problem, or when all scenarios of the Value Pack are in a "Failed" or "Degraded" state.

Degraded state: when the state of one or more scenario is "Degraded", and/or the mediation is not available.

Details of the possible operations depending on the status of the value pack are provided in the table below:

Value Pack state	Possible operations to execute on the Value Pack
n/a	Upload the Value Pack to the server
Not Deployed	Deploy the Value Pack Remove the Value Pack from the server
Stopped	Start the Value Pack Undeploy the Value Pack
Degraded	Stop the Value Pack Resynchronize (the mediation flows of) the Value Pack
Running	Stop the Value Pack Resynchronize (the mediation flows of) the Value Pack
Failed	Start the Value Pack Undeploy the Value Pack

Table 4 - Value Pack operations, depending on VP state

3.2.3 Scenario Level Operations

The scenario level operations are accessible either from the Value Pack Monitoring view or the Scenario Monitoring view.

The list of available operations on any given scenario is the following:

Possible Operation	Explanation
Reload (1)(2)	The "Reload" operation allows the UCA for EBC Server to reload a specific scenario without restarting the whole value pack. This may be required for example after changing some scenario rules or some scenario templates/parameters. When a Scenario is reloaded, the rules and template files are recompiled and the generated rules package is reloaded.
Clear WM	The "Clear Working Memory" operation clears the scenario's Working Memory. When that happens, all the facts are retracted from the scenario's working memory, except a few UCA for EBC system facts: Synchronization Flag Asynchronous Actions Flag Garbage Collection Flag Tick Flag Scenario Initialization Flag: this flag is present if it has been inserted by the rules Fire All Rules Flag: the flag is present only when the Scenario's "Fire All Rules" policy is set to WATCHDOG (instead of EACH ACCESS)

Possible Operation	Explanation
Dump WM	The "Dump Working Memory" operation dumps the content of the Working Memory (the list of facts in WM) into the UCA for EBC application log file and the scenario specific log file (if this log file is enabled). One log message is added to the log(s) for each fact in the scenario's Working Memory. Refer to [R1] UCA for Event Based Correlation - Reference Guide for more information on WM.
Reset Status	The "Reset Status" operation resets the scenario status back to "Running" in case the Scenario was in a "Degraded" state. This operation may be used if a non-fatal rules exception had caused the Scenario status to be "Degraded".

Rule Operation	Explanation
Remove Rule	The "Remove" operation available in the Rules list grid allows removing the rule from the knowledge base of the scenario running in UCA for EBC server.
Load Rule File	The "Load" operation available in the Rules files list grid allows compiling and loading the specified unloaded rules file in the scenario knowledge base.
Reload Rule File	The "Reload" operation available in the Rules files list grid allows compiling and reloading the specified rules file in the scenario knowledge base.
Unload Rule File	The "Unload" operation available in the Rules files list grid allows unloading the specified rules file from the scenario. As a consequence, the whole rules set (i.e. all the rules of that package) is unloaded from the scenario knowledge base.

Table 5 - Scenario and rules operations

Note

([1]) Changes to the filter files are not taken into account by the "reload" operation. Any change to the filters requires a full restart of the value pack.

([2]) There's no need to clear the Working Memory before reloading the rules, unless you want to start with both new rules and an empty Working Memory.

3.2.4 Additional administration tools

Additional administration tools are accessible from the **UCA-EBC:instanceName > Application > Tools** view:

Possible Operation	Explanation
Reload Logging Configuration File	The Logging mechanism for UCA for EBC Server is based on Log4J. Reloading the log4j configuration file forces UCA for EBC Server to take the new/updated Log4J configuration into account without stopping UCA for EBC Server.
Clear Log Database	The logs displayed at the UCA for EBC User Interface are stored in a database. This database requires regular cleanup in order to prevent the database file to grow indefinitely. The “Clean Log Database” operation deletes all log entries from the database.

Table 6 - Additional operations explained

Chapter 4

UCA for EBC Configuration

UCA for EBC User Interface brings ability to configure through the web interface the behavior of the server, the value packs and their scenarios. In particular, you can configure:

- the Application Action Registry
- the Value Packs scenario policies and mediation flows
- the Scenario specific configuration, along with its filters, mappers and templates

When navigating to a Configuration panel, by default, the panel is in view mode: that is, the configuration cannot be changed. It is intended for monitoring the configuration parameters.

Any configuration panel can be described as follows:

The left part of the configuration view is a tree browser for configuration items that lets you navigate through the configuration items data tree. Once a configuration item has been selected in the tree, the panel on the right hand-side shows the configuration parameters key/value pairs specific to the selected configuration item.

In order to edit and modify this configuration, you will have to click on the “Edit Configuration” icon of the toolbar available above the configuration tree browser. Then, the whole toolbar is displayed to allow:

- Undo / Redo last changes
- Cut / Copy / Paste an element from the configuration tree browser
- Add / Remove an element from the configuration tree browser
- Save / Apply changes to the server
- Refresh configuration from values stored in the server

The panel on the right-hand side displays elements and attributes of the XML element chosen in the configuration tree. You can modify/add/remove those elements and attributes depending on the schema delivered along with UCA for EBC (some elements are mandatory, etc...)

The following figure is an example of editing a scenario standard configuration

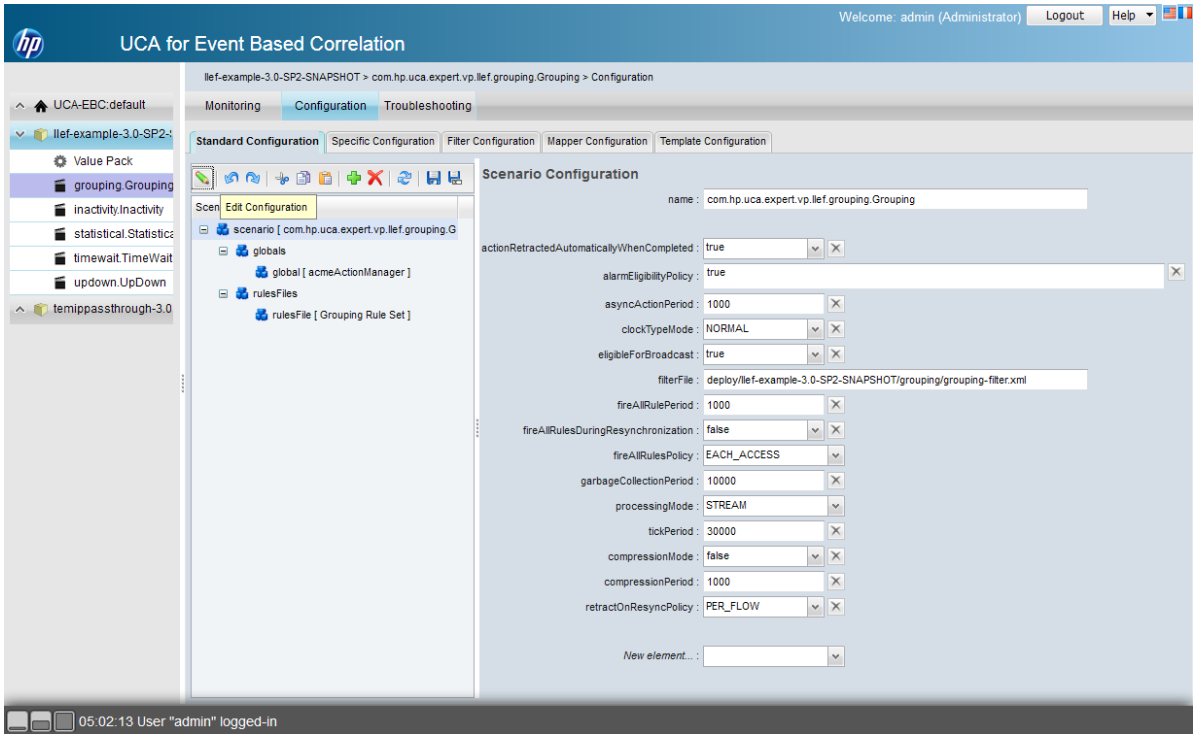


Figure 7 - Standard scenario configuration – Edit mode

4.1 Action Registry Configuration

Action registry is intended to configure mediation value packs, also known as channel adapters.

The Actions Registry standard configuration view is accessible from the ***UCA-EBC:instanceName*** **> *Actions* > Configuration > Standard Configuration** menu and is displayed as shown in the following screen capture:

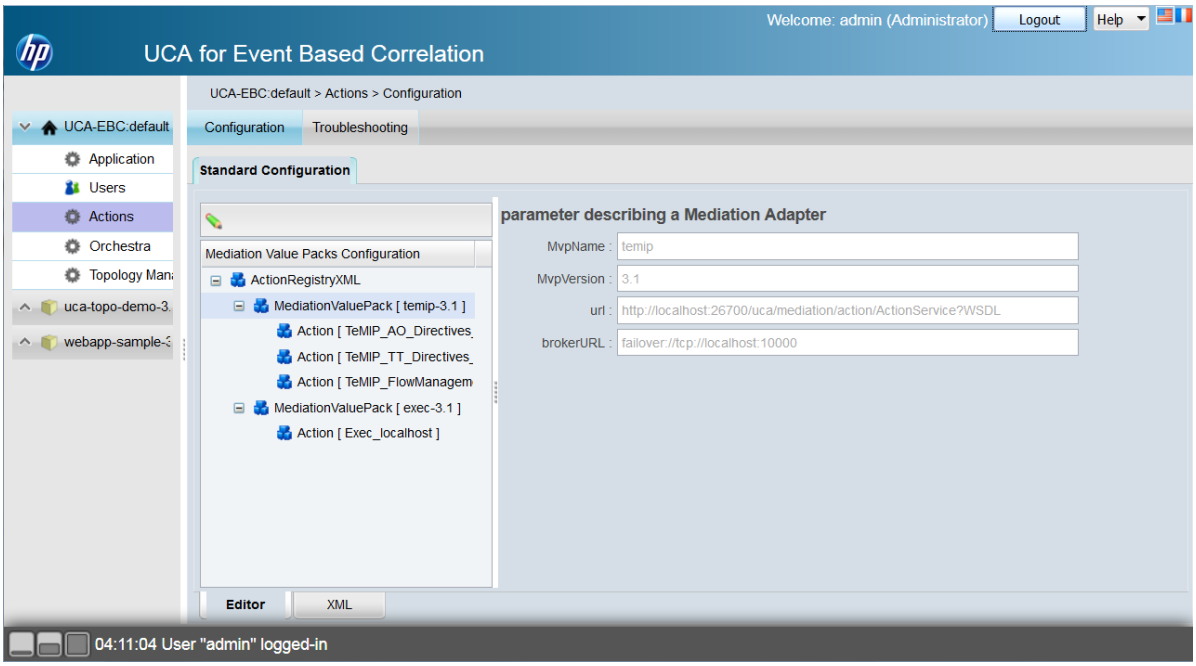


Figure 8 - Action Registry configuration

The modifications will be applied to **conf/ActionRegistry.xml** file.

Note

It is recommended that the UCA-EBC server has to be restarted in order for the changes in mediation value packs standard configuration to be properly taken into account.

4.2 Orchestration Configuration

Orchestrating event cascading is the capability of defining an event workflow between scenarios running on the same UCA for EBC Server instance. This workflow is made of several routes, each of them describing the way in which an event is cascaded to one or more other scenarios through the '`applyOrchestration()`' method.

The Orchestration configuration view is accessible from the ***UCA-EBC:instanceName > Orchestra > Configuration*** menu and is displayed as shown in the following screen capture:

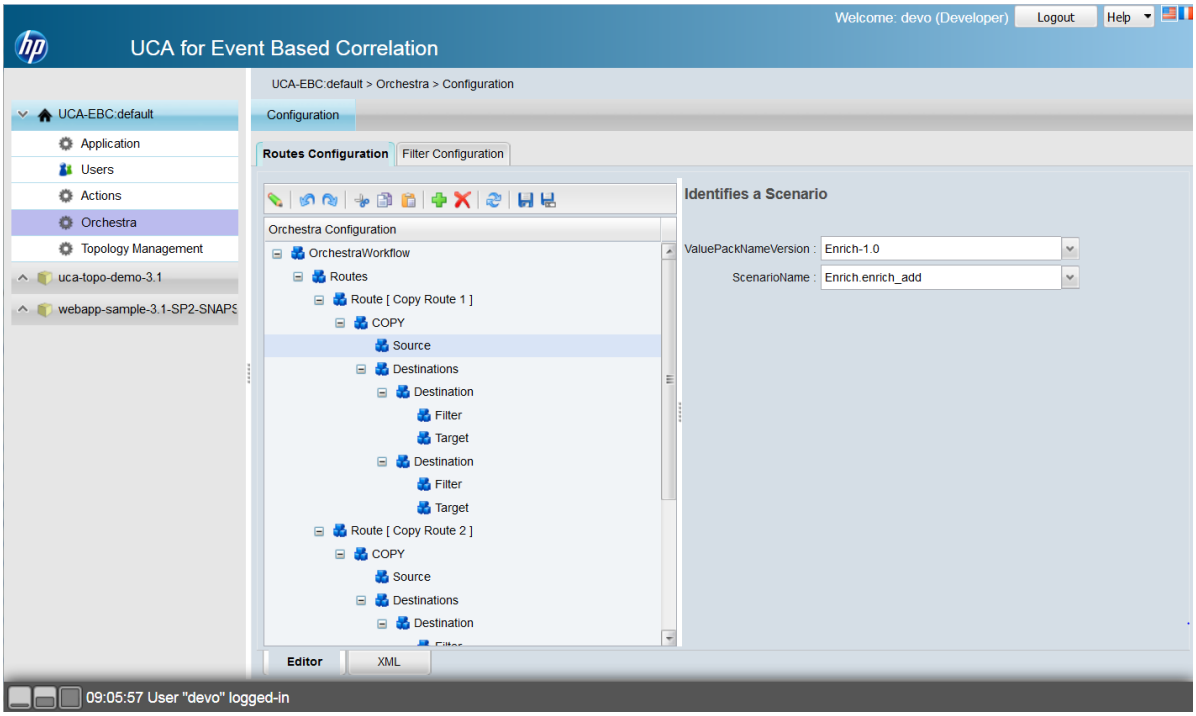


Figure 9 - Orchestration configuration

This view is made of two panels:

- Routes Configuration:

  The 'Routes configuration' panel allows for viewing/modifying the content of the `OrchestraConfiguration.xml` File

- Filter Configuration:

  The 'Filter Configuration' panel allows for viewing/modifying the content of the `OrchestraFilter.xml` file that defines the filters that can be used in route definitions.

The modification of these two configuration files can be performed with "developper" or "administrator" roles only.

Note

The UCA-EBC server instance has to be restarted in order for changes to the Orchestration files to be taken into account.

4.3 Value Pack Common Configuration

The *ValuePackConfiguration.xml* value pack configuration file stored under the *deploy/<valuePackName>/* directory can be edited from the GUI.

This file is made of several sections. Some are common to all scenarios of the Value pack (mediation flows and DB flows definitions), some other are specific to each scenario.

From this panel only the Value pack common sections (mediation flows and DB flows definitions) are accessible. The scenario specific part configuration is described in next chapter.

The Valuepack Common configuration view is accessible from the ***UCA-EBC:instanceName*** > ***Value Pack*** > **Configuration** menu and is displayed as shown in the following screen capture:

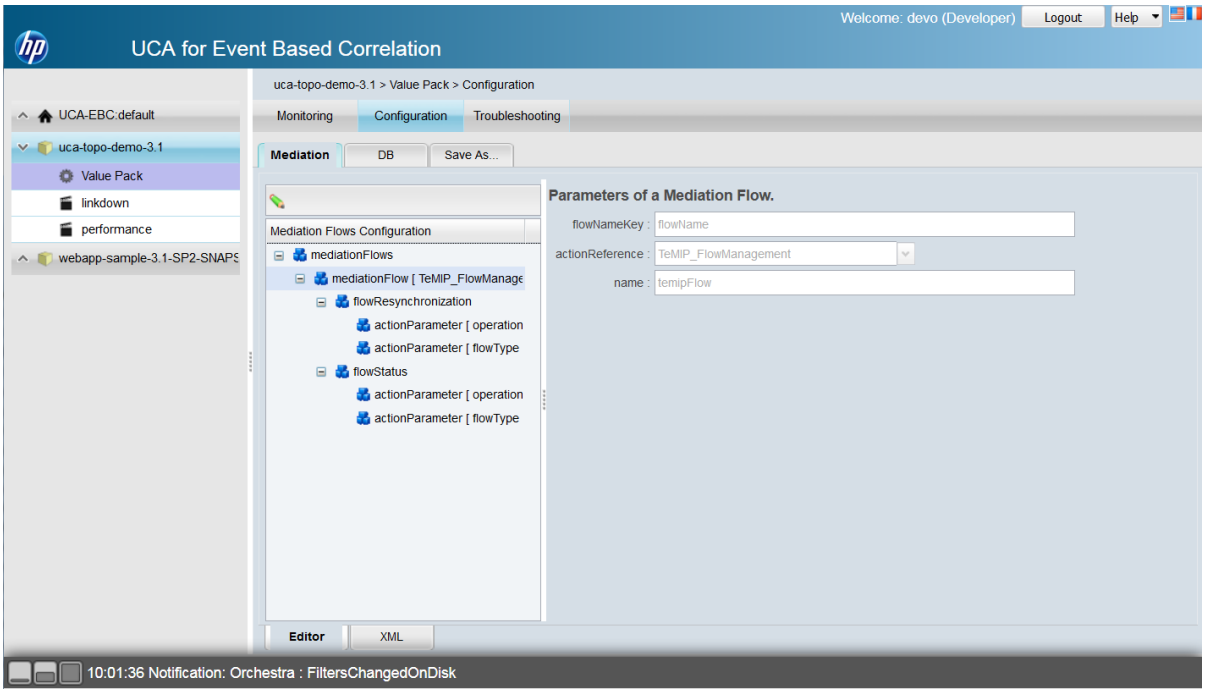


Figure 10 - Value Pack Mediation Flows configuration

As you can notice in above screen capture, you can **save as** your value pack configuration along with its binaries into a new value pack name-version.

Therefore, when changing a value pack (or scenario configuration), it is recommended to save as your value pack into a newer version, then un-deploy the current version, and then deploy the new version before making changes to it. In case of your configuration is completely messed up, you can still have the opportunity to redeploy your previous value pack name-version.

Note

The value pack standard configuration is visible **only** when value pack is deployed.

It is mandatory that the value pack has to be restarted in order for the changes in mediation flows standard configuration to be taken into account.

4.4 Scenario configuration

After any scenario configuration change, a value pack stop/start (restart) is recommended (necessary for some configuration parameters) for the changes to be taken into account.

4.4.1 Standard scenario configuration

The Standard scenario configuration view is accessible from the ***ValuepackName*** > ***scenarioName*** > **Configuration** > **Standard Configuration** menu and is displayed as shown in the following screen capture:

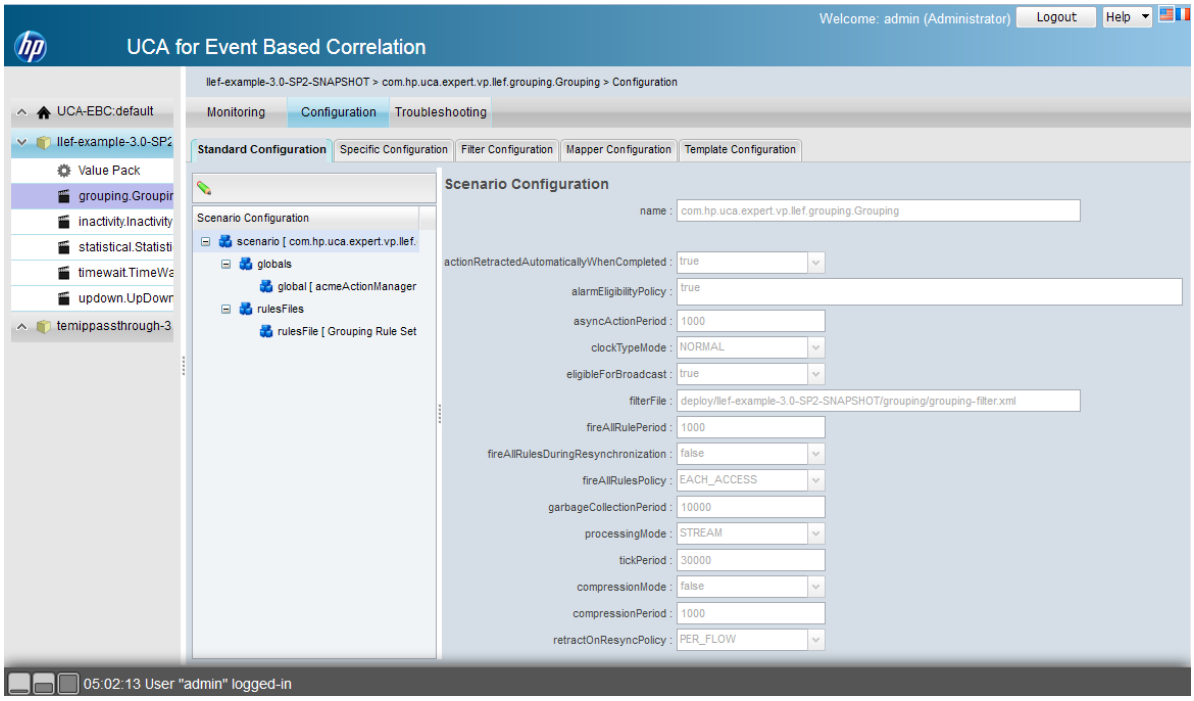


Figure 11 - Standard scenario configuration - View

The standard scenario configuration consists in a set of configuration parameters that are common to all scenarios. Each scenario has the same set of standard configuration parameters. However, each scenario has its own values for those standard configuration parameters.

☞ Please refer to the "*UCA for Event Based Correlation - Value Pack Development Guide*" for full details on the standard scenario configuration.

The standard configuration of a scenario defines the scenario policies. The modification will be applied to the **deploy/<valuePackName>/ValuePackConfiguration.xml** file.

Note

The scenario standard configuration is visible **only** when value pack is deployed.

It is mandatory that the whole Value Pack has to be restarted in order for the changes in standard scenario configuration to be properly taken into account.

4.4.2 Scenario-specific configuration

The Scenario-specific configuration view is accessible from the ***ValuepackName*** > ***scenarioName*** > **Configuration** > **Specific Configuration** menu.

The following screen capture is an example of some scenario-specific configuration for a scenario of the Problem Detection value pack.

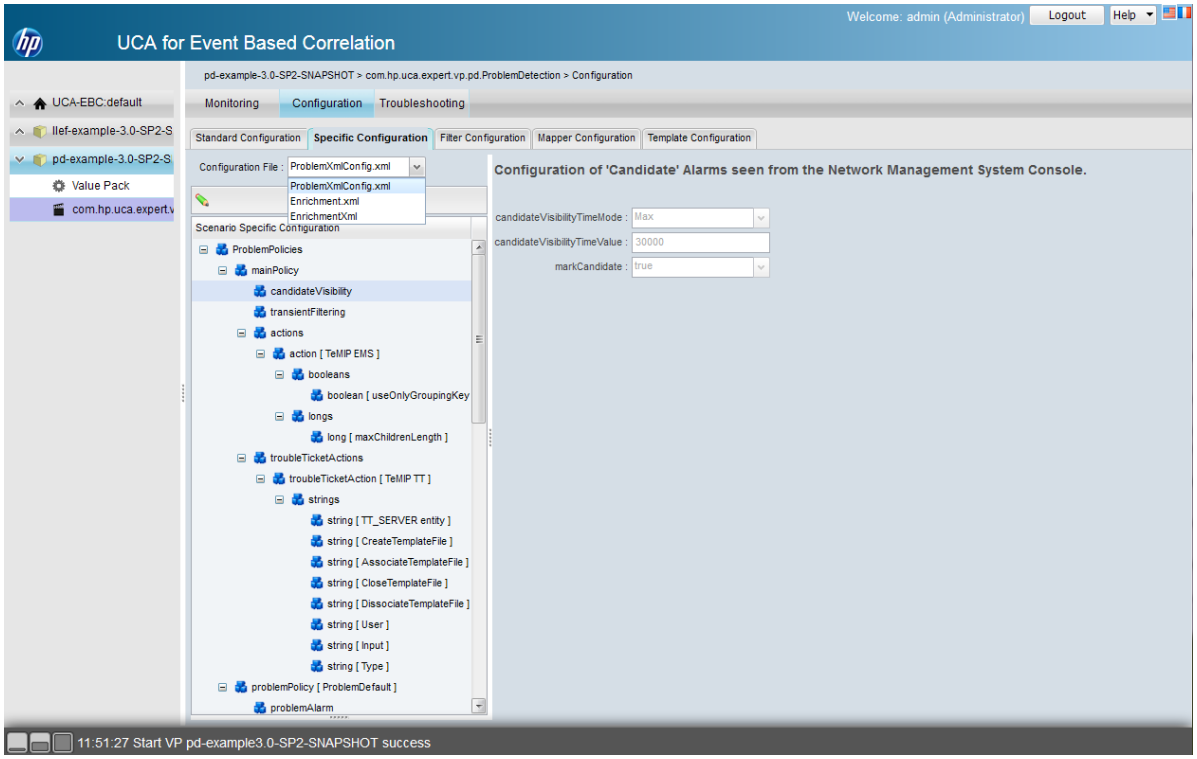


Figure 12 - Scenario-specific configuration

The scenario-specific configuration consists of a set of configuration values that are scenario dependent.

As for the standard scenario configuration, the left part of the scenario-specific configuration view is a tree browser for configuration items that lets you navigate through the configuration items data tree. Once a configuration item has been selected in the tree, the panel on the right hand-side shows the scenario-specific configuration parameters key/value pairs specific to the selected configuration item.

☞ Please refer to your Value Pack documentation for full details on any scenario-specific configuration.

Note

The scenario-specific configuration is visible **only** when value pack is started.

Above figure is an example of the ProblemXmlConfig.xml file coming with Problem Detection scenario.

4.4.3 Filter Configuration

The Filter Configuration tab exposes the content of the scenario filter XML file(s). If multiple filter files are defined for your scenario, you can select the filter file to display/edit using the “Filter File” drop-down menu. Reviewing the scenario filters can help investigate potential filtering problem for a scenario.

Below is a screen shot that shows the filter configuration for a scenario:

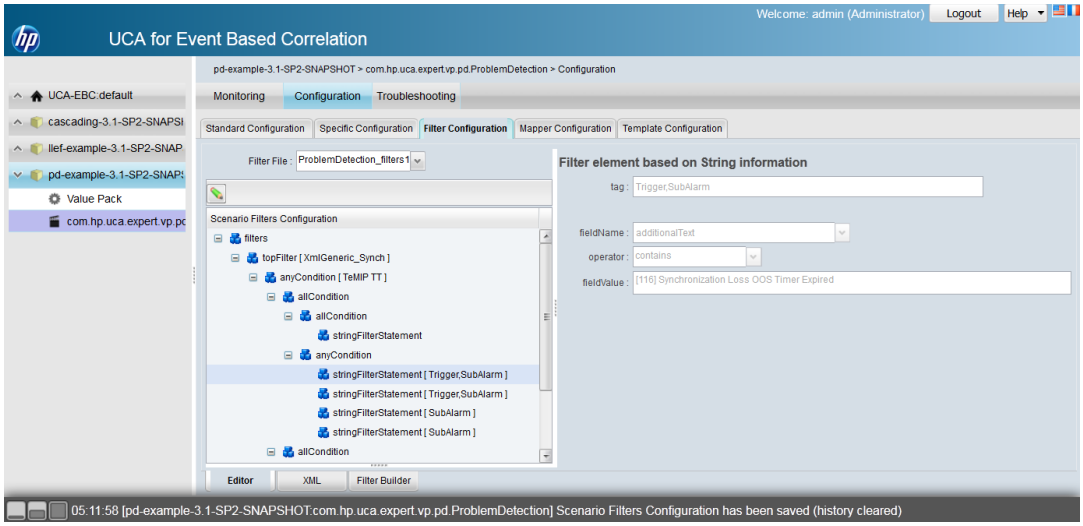


Figure 13 - Scenario Filter Configuration

Notes

On configuration change saved on the server, the whole value pack has to be restarted in order for the changes in filter configuration to be taken into account.

Multiple filter files per scenario are supported and can be directly modified from the UI.

4.4.3.1 Filter Tags Editor

In edition mode, if the value pack has been designed to support tags editing feature, a button will allow to launch the tag editor form:

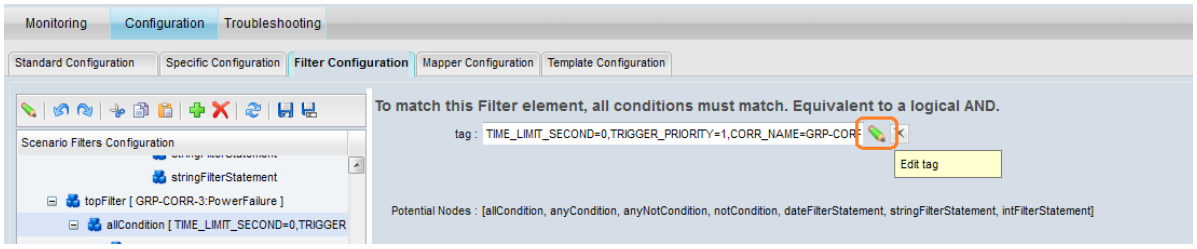


Figure 14 - Scenario Filter Tag Edit button

The Filter tag editor is an easy way to configure the tag field using well-known values that are defined by the value pack.

Below is a screen shot that shows the filter tag editor utility form:

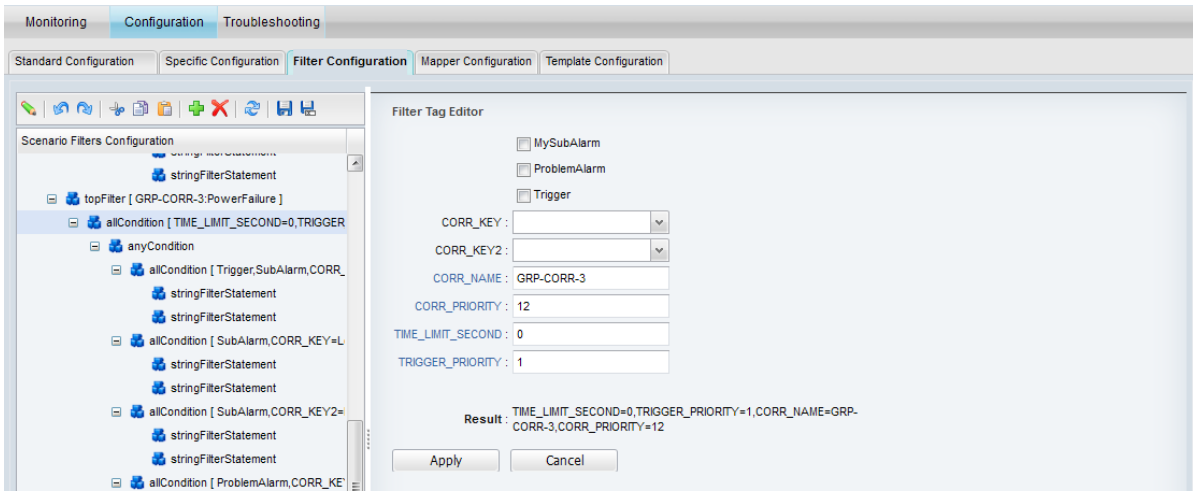


Figure 15 - Scenario Filter Tag Editor

Note

The Filter tag editor may disappear in next version as it is replaced by the Filter Builder Utility which is a more complete option to define tags, which can be grouped under different categories.

4.4.3.2 Filter Builder Utility

The Filter Builder panel is available under Filter Configuration. It allows a better view of existing filters displayed in a tabular mode and makes it easier to edit or create new filters.

It allows creating filters with following (mostly used) pattern:

```
<topFilter name="…" tagsGroup="…" >
      <anyCondition>
         <anyCondition tag="PATTERN_..." >
            <anyCondition tag="…" >
               <allCondition>
                  <notCondition> … </notCondition>
                  <stringFilterStatement> … </stringFilterStatement>
                  <intFilterStatement> … </intFilterStatement>
                  <dateFilterStatement> … </dateFilterStatement>
               </allCondition>
            </anyCondition>
            <anyCondition tag="…" >
               <allCondition> … </allCondition>
               <allCondition> … </allCondition>
            </anyCondition>
         </anyCondition>
         <anyCondition tag="PATTERN_..." >
   …
```

Above pattern makes it easy to classify your top filters and your “patterned” conditions that are part of them.

The first level anyCondition is a mandatory container for following anyConditions and is not displayed on the Filter Builder tool.

The second level anyCondition is used to contain your conditions. It is tagged with a PATTERN_ prefix to be easily retrieved. It is displayed as a contentPanel where title is the PATTERN_... tag. From this anyCondition, you can add multiple anyCondition under it.

The third level anyCondition is used to contain the actual tags that apply for the conditions part of it. Those conditions are stored in an allCondition element that is displayed as a grid form on the Filter Builder tool. Such a grid contains statements (which can be strings, integers or dates) and also notCondition statements. Each kind of statement is represented by a different icon in the grid. You can of course have multiple allConditions under a third level anyCondition, but all those allConditions will have the same tags set.

The following screenshot is an example:

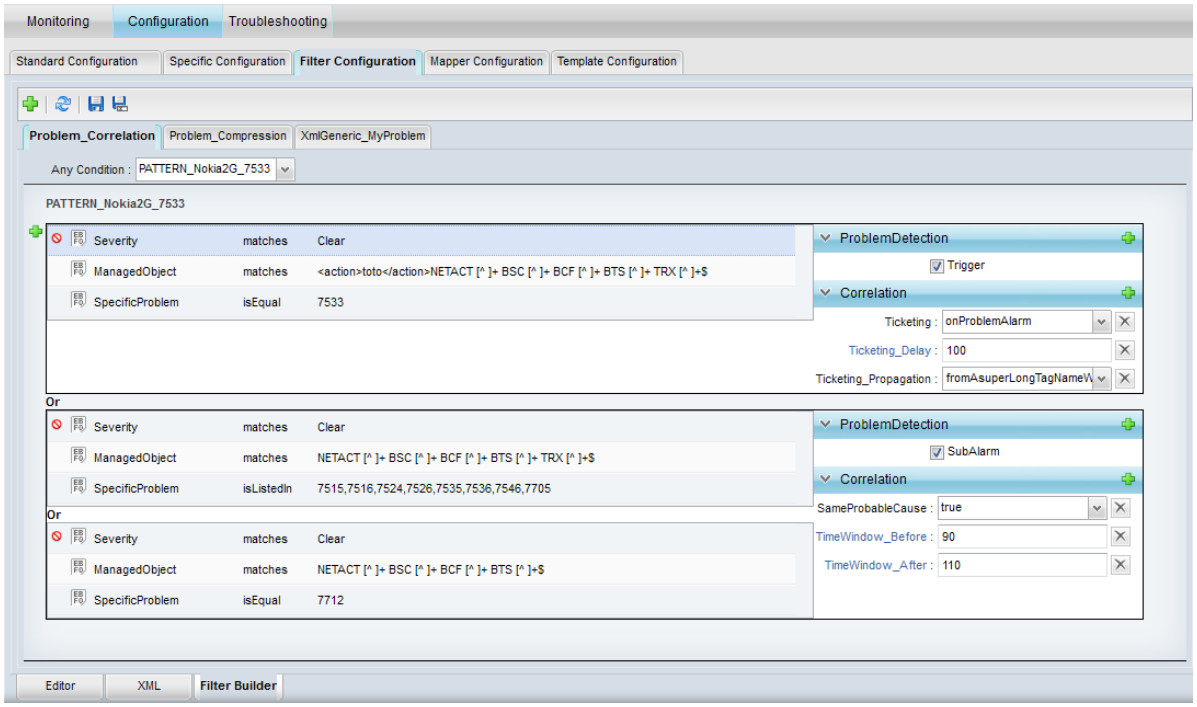


Figure 16 - Scenario Filter Builder Tool

Each top filter is displayed within its own panel (multiple tabs above toolbar).

Toolbar at the top is for:

- Adding a new top filter
- Refreshing filters from server file
- Saving filters to server file

Most of other actions are available through mouse right-click. A mouse right-click is dependent on the region where it is performed. This includes:

- Removing a top filter
- Adding/Removing a PATTERN_ condition
- Copy/Paste a PATTERN_ condition
- Adding/Removing tagged condition under a PATTERN_ condition
- Adding/Removing an allCondition under a tagged condition

- Edit/Remove Statement part of an allCondition
- Adding/Removing String/Date/Int Statement part of an allCondition
- Set/Unset a Statement as a notCondition
- Adding/Removing a predefined tag

Furthermore, there is a strict control on tag edition:

- A parameter tag can now have a default value from enumerated values
- A parameter tag can have a type (int/boolean/string/enum)
- A parameter tag can be set as required

4.4.4 Mapper Configuration

The Mapper Configuration tab exposes the content of the Mapper definition files as shown in the following screenshot:

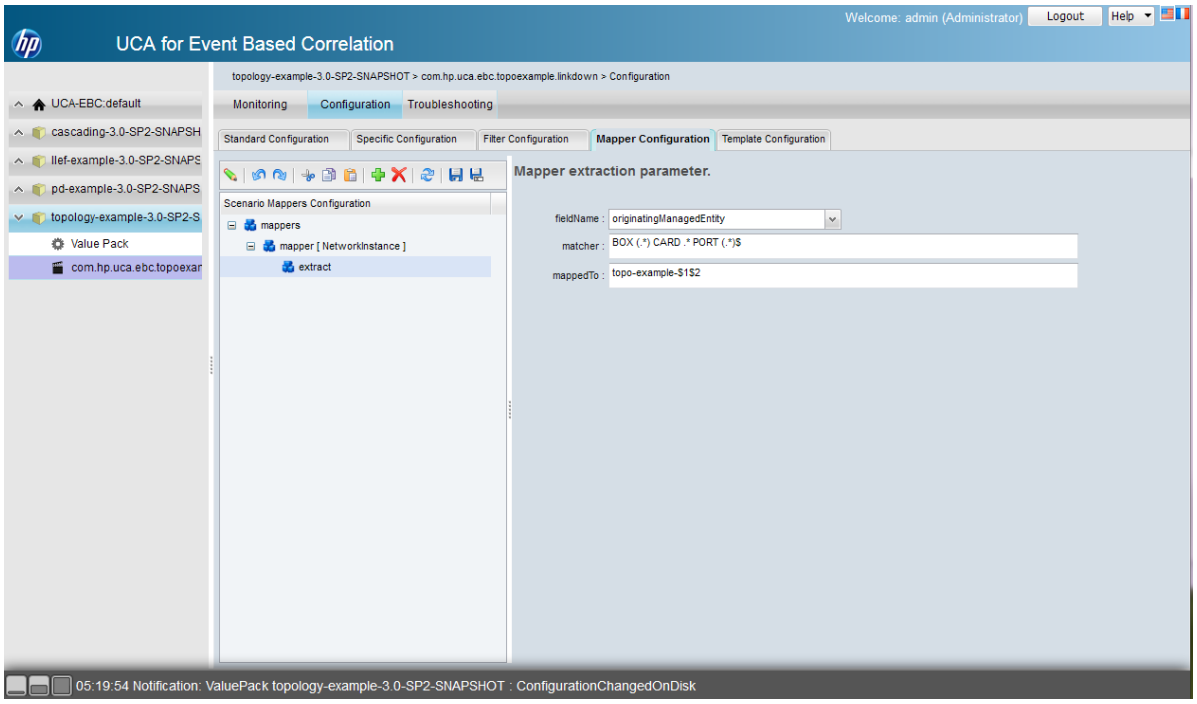


Figure 17 - Scenario mapper configuration

4.4.5 Template Configuration

The Template Configuration tab exposes the content of the template definition files as shown in the following screenshot:

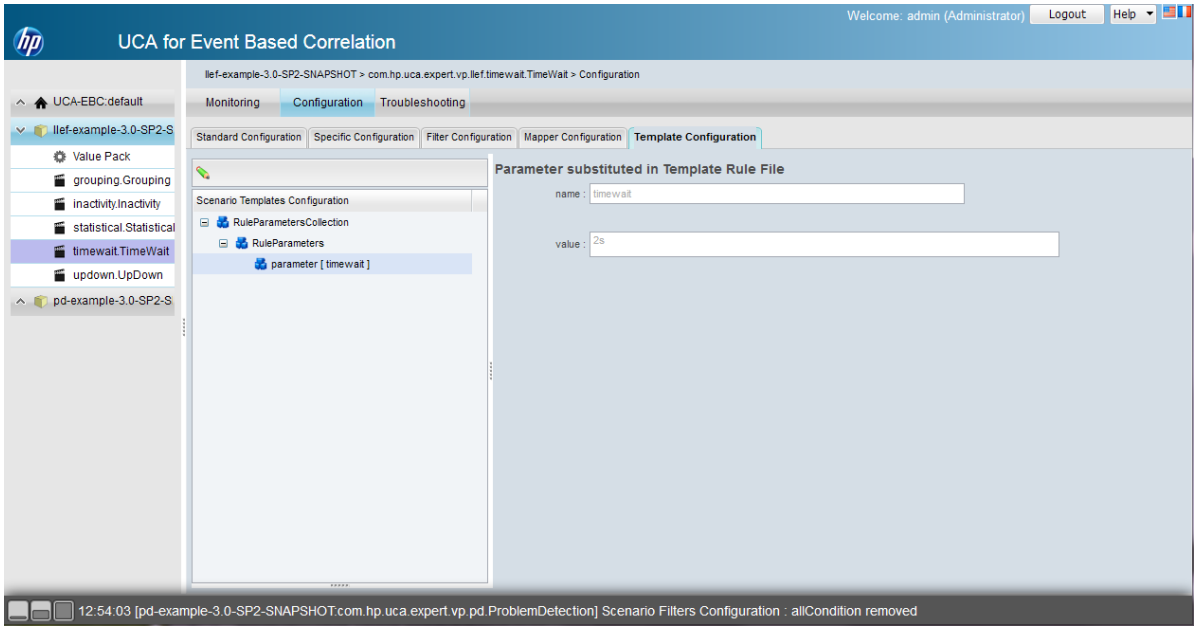


Figure 18 - Scenario template configuration

At the top of the template configuration content area, in the only case where there are more than one template file associated with the scenario, a drop-down menu lets you select the template file for which you want to display the configuration,.

Once a template file has been selected, and in the same way as for the standard scenario and scenario-specific configuration, the left part of the template configuration view is a tree browser for configuration items that lets you navigate through the configuration items data tree. Once a configuration item has been selected in the tree, the panel on the right hand-side shows the template configuration parameters key/value pairs specific to the selected configuration item.

Note

Upon template configuration change saved on the server, the whole value pack has to be restarted in order for the changes in template configuration to be taken into account.

Chapter 5

UCA for EBC Monitoring

The UCA for EBC User Interface provides monitoring capabilities to the UCA for EBC application at different levels:

Monitoring level	Explanation
Application Monitoring	The main monitoring level is the Application Monitoring. It displays the status of the application itself and an overview of the Value Pack status.
Value Pack Monitoring	The Value Pack monitoring level gives the value pack status as well as the status of all the scenarios and all the mediation flows defined in the Value Pack.
Scenario Monitoring	The Scenario monitoring level gives a monitoring view for the scenario and lists the rules names along with the rules files involved in the implementation of the scenario.

Table 7 - UCA for EBC User Interface monitoring levels

5.1 Application monitoring (or Dashboard)

This is the default view displayed when the UCA for EBC User Interface is launched.

The corresponding menu selection for Application Monitoring is the following:

UCA-EBC:instanceName > Application > Monitoring

The Application Monitoring panel, inside the content area, is made of two sections: the “UCA for EBC Status” section and the “Value Packs Status” section as shown in the screen capture below:

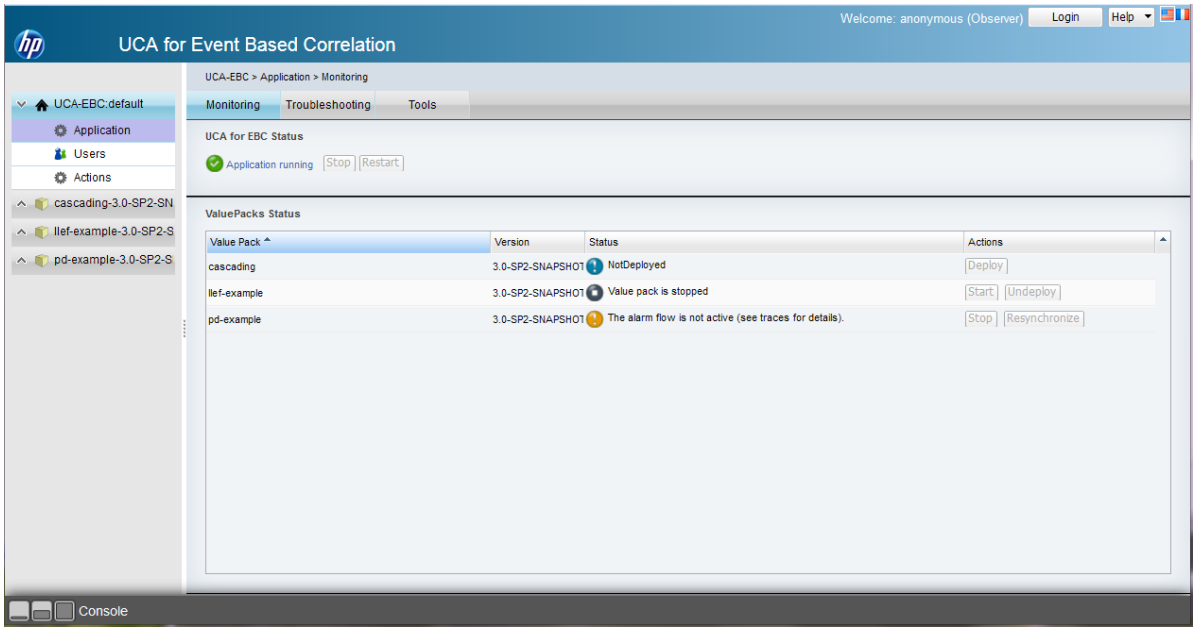


Figure 19 - UCA for EBC Application Monitoring View (or Dashboard)

The "UCA for EBC Status" section gives a quick status of the application. Usually the status of the application is "Running", which means that the application is running properly. Remember that if the UCA for EBC Server is not running at all, then your web browser will not be able to connect to it and therefore nothing will be displayed.

The Value Packs Status section contains a table giving the status of all the Value Packs installed on UCA for EBC.

The status for each Value Pack can be one of the following:

Value Pack Status	Explanation
Not Deployed	This indicates that the Value Pack is present in the `$UCA_EBC_HOME/valuepacks` directory but has not been deployed.
Stopped	This indicates that the Value Pack has been deployed but is not actually started yet.
Running	This indicates that the Value Pack has been deployed and started successfully and all scenarios are working fine.
Degraded	This indicates that the Value Pack is running but some of the scenarios did not start properly and are either in the 'degraded' or 'failed' state.
Failed	This indicates that the Value Pack did not start correctly and is not working.

Table 8 - Value Pack Statuses

The "status" column in the Value Packs Status table gives additional details, especially if the Value Pack is in a "Degraded" or "Failed" state.

The rows in the Value Packs Status table are “double-click” sensitive. A double click on a row in the table automatically triggers a jump to the Value Pack Monitoring view for the corresponding value pack:

valuepack name > Valuepack > Monitoring

5.2 Value Pack monitoring

The Value Pack monitoring view is reached by selecting one of the value packs in the left-hand side main menu, then selecting “ValuePack” as the sub-menu item and finally selecting “Monitoring” on the top horizontal sub-menu:

For example, selecting **llef-example-3.3 > ValuePack > Monitoring** displays the Value Pack Monitoring view of the llef-example-3.3 value pack in the content area of the UCA for EBC User Interface as shown in the screen capture below:

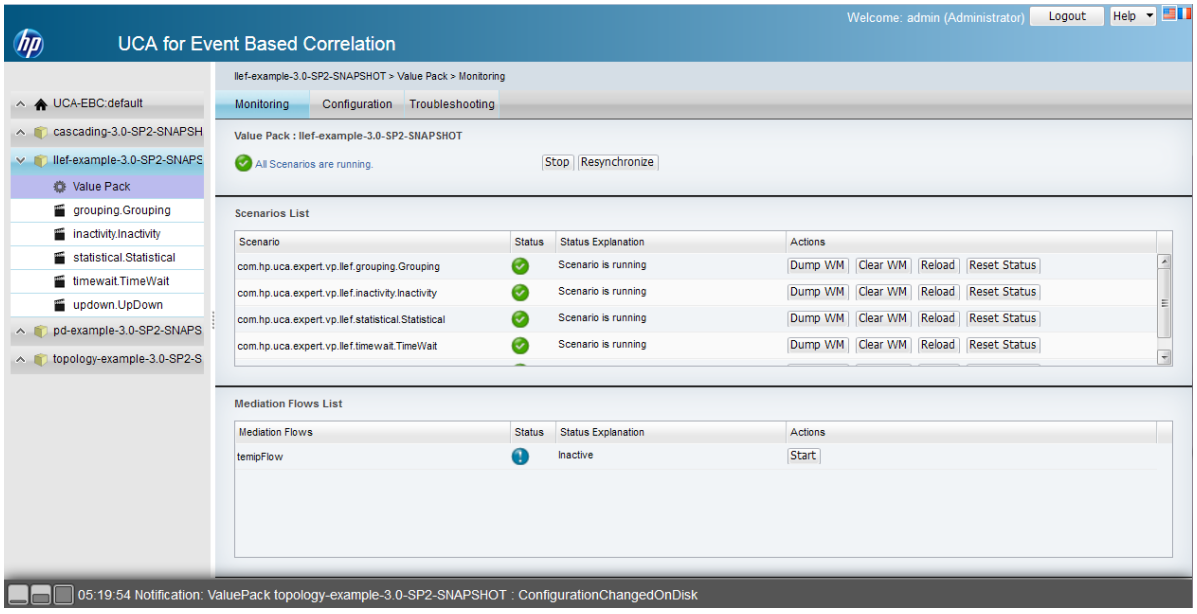


Figure 20 - UCA for EBC Value Pack Monitoring View

The Value Pack Monitoring panel is made up of three sections: the “Value Pack Status” section, the “Scenario List” section and the “Mediation Flows List” as shown in the above screen capture.

The “Value Pack Status” section gives a quick status of the Value Pack along with a detailed Value Pack status description in case the value pack is in a “Failed” or “Degraded” state.

The “Scenario List” section is composed of a table giving the status of each scenario within the Value Pack.

The status of each scenario can be:

Scenario Status	Explanation
Running	This indicates that the scenario is working properly.

Degraded	This indicates that the scenario has been started but an exception was raised at some point in time. Usually a "degraded" scenario is not working anymore.
Failed	This indicates that the scenario failed to start. This is usually due to a scenario configuration problem.
Stopped	This indicates that the scenario is currently not running (just correctly deployed)

Table 9 - Scenario Statuses

In case of "Failed" or "Degraded" status, the status explanation column in the Scenarios List table gives useful information for troubleshooting and correcting the problem.

The rows in the Scenarios List table are "double-click" sensitive. A double click on a row in the table automatically triggers a jump to the Scenario Monitoring view for the corresponding scenario:

valuepack name > the scenario > Monitoring

The "Mediation Flows List" section displays the status of each mediation flow which is defined within the Value Pack.

The status of each mediation flow can be:

Scenario Status	Explanation
Running	This indicates that the scenario is working properly.
Degraded	This indicates that the scenario has been started but an exception was raised at some point in time. Usually a "degraded" scenario is not working anymore.
Failed	This indicates that the scenario failed to start. This is usually due to a scenario configuration problem.
Stopped	This indicates that the scenario is currently not running (just correctly deployed)

Table 10 – Mediation Flows Statuses

5.3 Scenario monitoring

The scenario monitoring view gives two kinds of information:

A reminder of the status of the scenario: “Running”, “Degraded”, or “Failed”

The list of rules involved in the scenario implementation. The rules are also grouped in a “Rules Files List” panel in order to have actions available for that whole rules set.

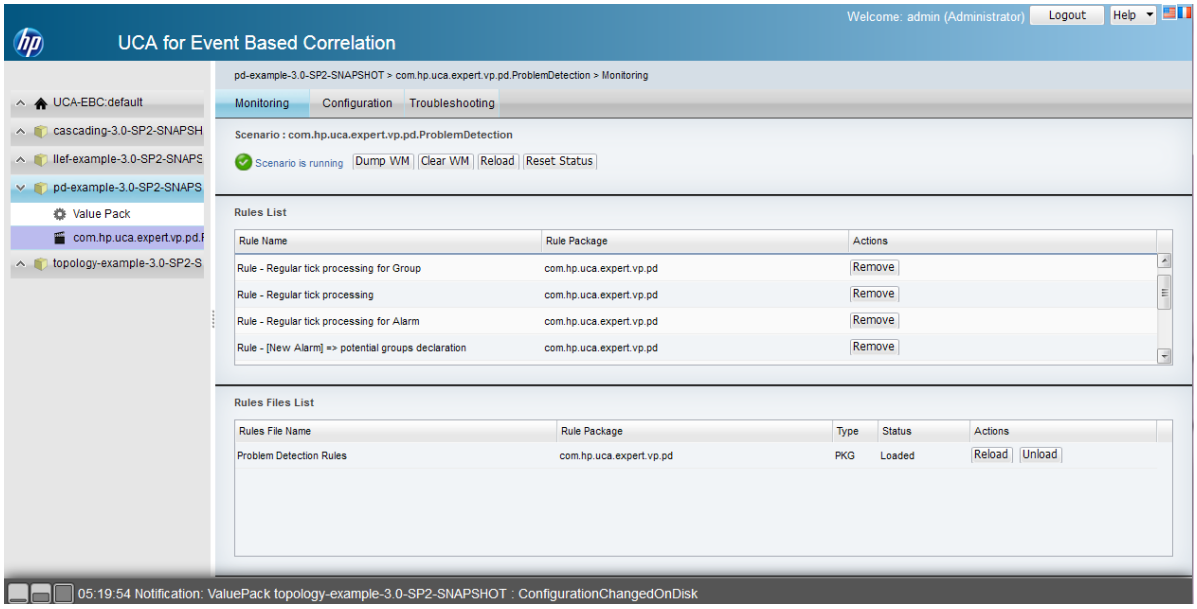


Figure 21 - UCA for EBC Scenario Monitoring View

Note

It is strongly recommended to have one rule package per rules file, because Drools manages the rules according the package name.

Chapter 6

UCA for EBC Troubleshooting

This chapter describes how the UCA for EBC User interface can be used for troubleshooting the UCA for EBC application itself, value packs or scenarios.

6.1 Monitoring internal statistics

The term “statistics” is to be understood as any collection of statistical data providing understanding of the internal behavior of the UCA for EBC application. Some examples of statistics are:

- Number of alarms collected
- Dispatching rate
- Internal queue size

Statistical data is retrieved from the UCA for EBC Server and delivered to the UCA for EBC User Interface every 5 seconds. Statistical data can be of the following data types:

- String
- Date
- Boolean
- Numeric

Numeric values can be displayed as graphs, so that their evolution over a period of time can be shown. Graphs can be displayed for any statistics in numeric format, simply by clicking on the “graph” icon () located to the right of the numeric value.

Internal statistics can be monitored at several levels. The following is a screenshot of the Statistics content area at the Application Level:

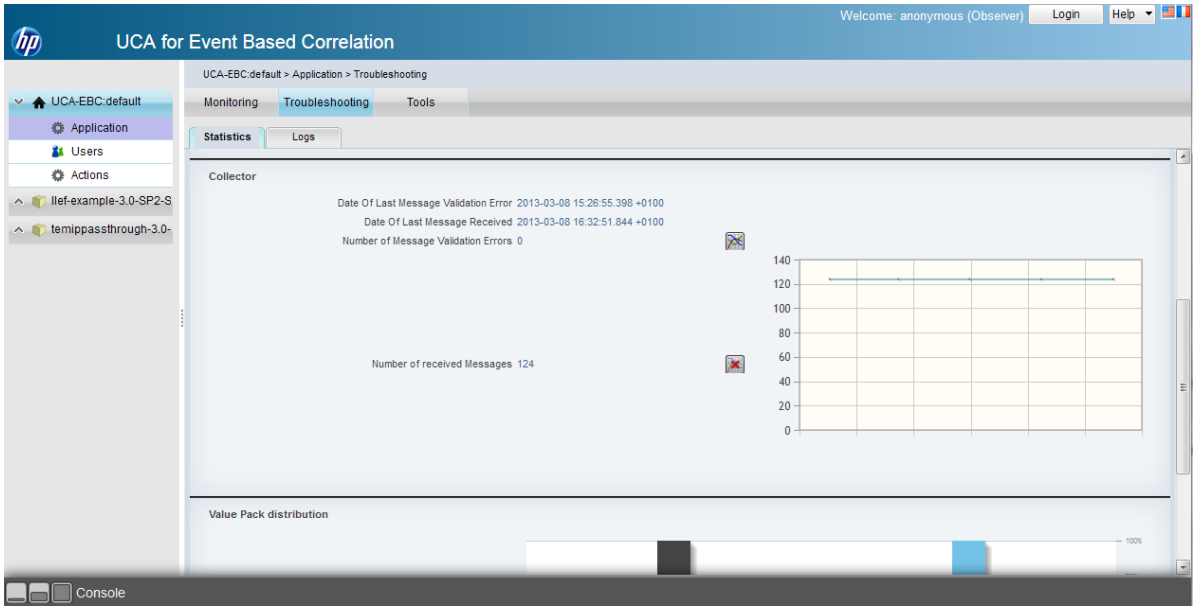


Figure 22 - Statistics content area at Application Level

For Application level statistics, please go to the following menu:

UCA-EBC:instanceName > Application > Troubleshooting > Statistics

For Action level statistics, please go to the following menu:

UCA-EBC:instanceName > Actions > Troubleshooting > Statistics

For Value pack level statistics, please go to the following menu:

valuepack name > Valuepack > Troubleshooting > Statistics

For Scenario level statistics, please go to the following menu:

valuepack name > *scenarioname* > Troubleshooting > Statistics

Below is a table that describes the kind of statistical data collected at each level:

Level	Statistical data collected
Application level	At this level, statistics regarding two main internal components of the UCA for EBC application are collected: Statistics on the collector Statistics on the dispatcher Statistical data on the collector gives information on the number of messages collected from the UCA for EBC input queue, the date of the last inbound message and the number of validation errors for inbound messages. Statistical information on the dispatcher gives information on the number and the type of messages that were dispatched to value packs. Information on the dispatcher's queue is also collected, which may be useful to monitor the dispatcher's behavior in case of bursts of inbound messages.
Action level	The action level represents the "down-stream" flow from UCA for EBC to the mediation layer. Action execution requests from scenarios are pushed to UCA for EBC action queue which in turn

Level	Statistical data collected
	sends them down to the mediation. The statistical data collected at the Action level is mainly related to the Action queue.
Value pack level	Statistical data collected at the Value Pack level data monitors the alarms that have been dispatched to a Value Pack since it was first started: number of alarms date of last alarm dispatched to the value pack percentage of alarms received by UCA for EBC actually dispatched to the value pack
Scenario level	Statistical data collected at the Scenario level data monitors the scenario behavior. Scenario statistical data is split into 3 sections: the “filter” section the scenario collection queue section the working memory section The “filter” section gives information on the number of events passing through the filter or being rejected by it. The scenario collection queue section gives information on the rate at which the scenario is capable of consuming events: this rate can be monitored by checking: the queue size the date and time of the last high water mark of the queue the date and time of the last event (Alarm) added to the queue the date and time of the last event (Alarm) removed from the queue to be processed by the Scenario the date and time of the last time the queue was empty the high water mark of the queue whether the high water mark is currently increasing or not the number of times the queue was empty since the last high water mark the total number of objects added to the queue since start-up the total number of objects added to the queue since the last high water mark Finally, the working memory section gives information on the rule engine working memory associated with the scenario: The current number of facts (objects) in Working Memory The rate of Insertion/Update/Deletion of the Working Memory (in operations per second) The maximum number of facts in Working Memory since start-up The number of facts inserted in the Working Memory since start-up The number of facts retracted from the Working Memory since start-up

Level	Statistical data collected
	The number of facts updated in the Working Memory since start-up
	The date and time of the last fact inserted in the Working Memory
	The date and time of the last fact retracted from the Working Memory
	The date and time of the last fact updated in the Working Memory
	A flag (true/false) indicating whether a mediation flow is currently in the middle of a synchronization or not

Table 10 - Statistics collected by level

Note

The same statistics can be monitored through JMX using the Java console connected to UCA for EBC Server.

☞ Please refer to the "*HP UCA for Event Based Correlation - Administration, Configuration and Troubleshooting Guide [R3]"* for more information on how to view the UCA for EBC Server statistics using the Java JMX Console.

6.2 Displaying application logs

UCA for EBC application logs are important for understanding how the application behaves or investigating problems, especially during the integration phases.

The UCA for EBC application logs are produced by the UCA for EBC Server using the log4j technology. The logs created though the log4j appender named "DB" can be browsed through the UCA for EBC User Interface.

Alternatively, UCA for EBC application administrators can configure the log4j configuration file in order to use some external tools (such as Chainsaw for example) to browse the logs.

In any case, the UCA for EBC User Interface provides an efficient and easy way to browse the logs.

The logs are available at several levels:

- Application level logs
- Value pack level logs
- Scenario level logs

Application level logs are accessible from the following menu:

UCA-EBC:instanceName > Application > Troubleshooting > logs

Value pack level logs are accessible from the following menu:

valuepack name > Valuepack > Troubleshooting > logs

Scenario level logs are accessible from the following menu:

valuepack name > *Scenario name* > Troubleshooting > logs

The following screenshot shows Troubleshooting/Log panel at application level:

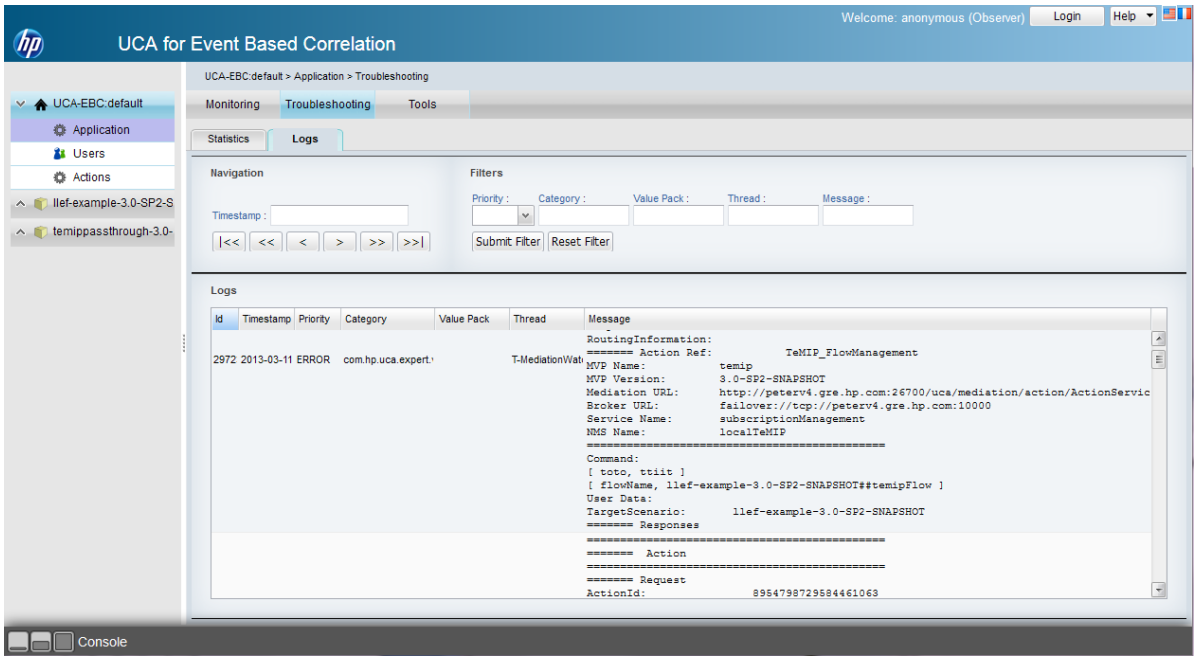


Figure 23 - Troubleshooting/Log panel at Application level

Regardless of the level, the Troubleshooting/Log panel is made up of three sections:

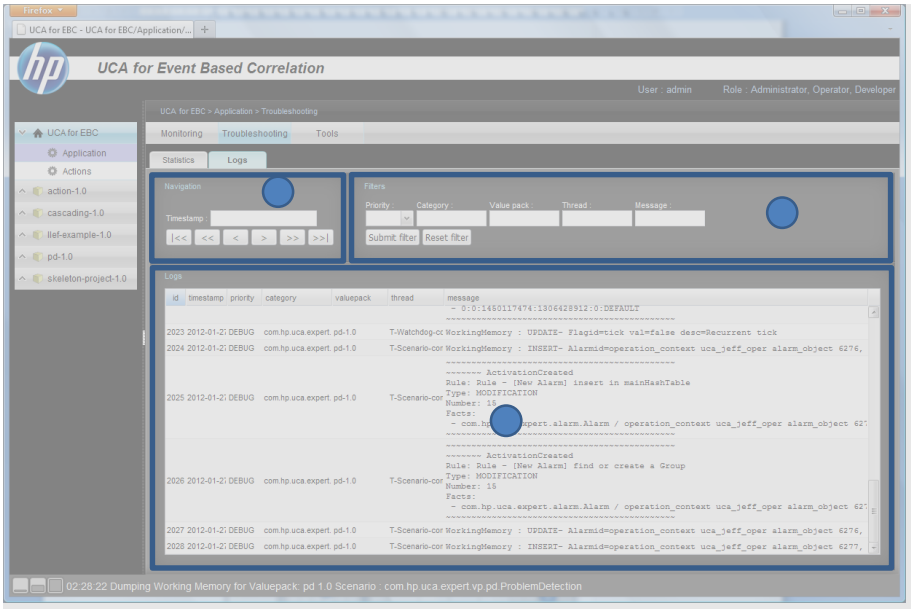


Figure 24 - Troubleshooting/Log panel layout

Section 1: Section 1 is the player section. It provides navigation throughout the log file in any direction (forward or backward) using the following buttons:

|<< move back to the start of the log

<< move back 10 screens

< move back 1 screen

> move forward 1 screen

>> move forward 10 screens

>>| move forward to the end of the log.

Another way to move to a specific location in the log file is to specify a timestamp. Entering a timestamp with the form: HHHH-MM-DD HH.MM.SS.mmm will navigate to the next log after this timestamp.

Section 2: Section 2 is the filter section. You can define filters on the logs depending on several criteria such as: Priority, Category, Thread, Value Pack.

Filtering can be made even more specific by specifying a substring to be matched against the log messages.

Two buttons are also available to:

Submit: Submit filter changes (and apply them)

Reset: Clear all the filters.

Section 3: Section 3 is the log content area which displays the list of log messages that passed the filter.

UCA for EBC optional displays

This chapter describes which of the UCA for EBC User interfaces are optionally displayed.

6.3 Topology Management

The sub-menu **Topology Management** is displayed only when Neo4J database has been configured for UCA-EBC.

Within this sub-menu, you will have 3 content areas:

For Topology display, please go to the following menu:

UCA-EBC:instanceName > Topology Management > Display

For getting the administration of Neo4J, please go to the following menu:

UCA-EBC:instanceName > Topology Management > Topology Mgr

For Topology data load, which requires admin rights, please go to the following menu:

UCA-EBC:instanceName > Topology Management > Data load

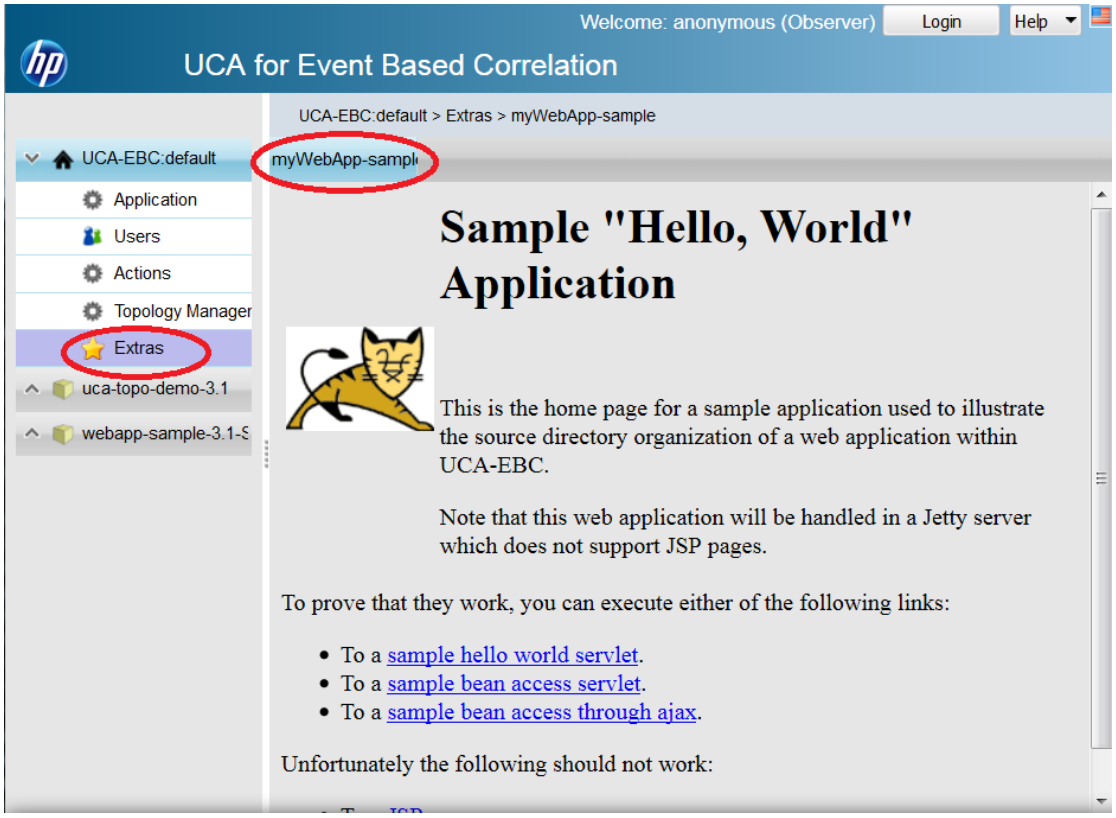
6.4 Extras

The sub-menu Extras is displayed when you have optionally put some extra .war files under the $UCA_EBC_INSTANCE/webapps directory. This directory is optional and is not created by default.

Each .war file stored in this directory will be displayed by UCA for EBC UI under the following menu:

UCA-EBC:instanceName > Extras > <name of .war file>

As shown in the picture below:


Welcome: anonymous (Observer)
Login
Help
UCA for Event Based Correlation
UCA-EBC:default > Extras > myWebApp-sample
UCA-EBC:default
myWebApp-sampl
Application
Users
Actions
Topology Manager
Extras
uca-topo-demo-3.1
webapp-sample-3.1-S
Sample "Hello, World" Application
This is the home page for a sample application used to illustrate the source directory organization of a web application within UCA-EBC.
Note that this web application will be handled in a Jetty server which does not support JSP pages.
To prove that they work, you can execute either of the following links:
To a sample hello world servlet.
To a sample bean access servlet.
To a sample bean access through ajax.
Unfortunately the following should not work:

Glossary

UCA: Unified Correlation Analyzer

EBC: Event Based Correlation

DNS: Domain Name Service

IP: Internet Protocol

LOG4J: Standard Logging Mechanism for Java-based programs

URL: Uniform Resource Locator (identifies the location of a resource on the Internet)

WM: Working Memory of a scenario, which contains all the facts for this scenario.