

---

# HP Unified Mediation Bus



## Unified Mediation Bus

**Version 1.0**

## Installation and Configuration Guide

**Edition: 1.1**

**For the HP-UX (11.31), Linux (RHEL 6.5) and Windows® Operating Systems**

**October 2015**

© Copyright 2015 Hewlett-Packard Development Company, L.P.

---

## 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

© Copyright 2015 Hewlett-Packard Development Company, L.P.

### 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 2012®, Windows XP®, and Windows 7® 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.

Hazelcast™ is a trademark of Hazelcast Inc.

Apache Kafka™ is a trademark of the Apache Software Foundation.

Apache ZooKeeper™ is a trademark of the Apache Software Foundation.

Apache Bigtop™ is a trademark of the Apache Software Foundation.

# Contents

<b>Preface .....</b>	<b>7</b>
<b>Chapter 1.....</b>	<b>9</b>
<b>Introduction .....</b>	<b>9</b>
<b>Overview .....</b>	<b>9</b>
<b>Chapter 2.....</b>	<b>11</b>
<b>Unified Mediation Bus Server .....</b>	<b>11</b>
2.1 Licensing .....	11
2.2 UMB Server configurations overview.....	11
2.3 Pre-installation tasks .....	12
2.3.1 Java.....	12
2.3.2 Create UMB administration User 'hpossadm'.....	12
2.4 UMB zookeeper installation.....	13
2.4.1 Disk requirements .....	13
2.4.2 UMB Server zookeeper package installation .....	13
2.4.3 Un-installation of UMB Server ZooKeeper package .....	19
2.5 UMB Kafka installation.....	19
2.5.1 UMB Server kafka package installation .....	19
2.5.2 Un-installation of UMB Server Kafka package.....	25
2.6 Configuring UMB server for production environment .....	25
2.6.1 Configuring ZooKeeper in a Production Environment .....	25
2.6.2 Make the Kafka Broker Redundant .....	27
<b>Chapter 3.....</b>	<b>28</b>
<b>Unified Mediation Bus Adapters .....</b>	<b>28</b>
3.1 Licensing .....	28
3.2 Installation pre-requisites .....	28
3.2.1 Java.....	28
3.2.2 Admin user creation .....	29
3.3 Unified Mediation Bus Runtime Package installation.....	29
3.3.1 Product installation.....	29
3.3.2 Disk requirements for the UMB runtime .....	30
3.3.3 Files organization .....	31
3.3.4 Setting the Unified Mediation Bus Runtime environment variable (Unix only) ...	31
3.3.5 Un-installation of a Unified Mediation Bus Runtime.....	32
3.4 Unified Mediation Bus Adapters installation .....	32
3.4.1 Product installation.....	32
3.4.2 Disk requirements for an UMB adapter .....	33
3.4.3 Files organization .....	33
3.4.4 Ports used.....	34
3.4.5 Setting the Unified Mediation Bus Adapter environment variables (Unix only) ..	34
3.4.6 Starting and stopping a Unified Mediation Bus Adapter .....	35
3.4.7 Un-installation of a Unified Mediation Bus Adapter .....	35

3.5	Unified Mediation Bus Adapters configuration .....	35
3.5.1	AdapterConfiguration.xml file .....	35
3.5.2	adapter.properties file .....	36
3.5.3	hazelcast.xml file .....	41
3.5.4	log4j.xml file.....	43
3.6	TeMIP Adapter Specific configuration.....	44
3.6.1	TeMIP Adapter Installation.....	45
3.6.2	TeMIP Adapter Configuration .....	45
3.7	OSSAF Adapter .....	54
3.7.1	Specific properties.....	54
3.7.2	Specific configuration.....	54
3.7.3	Deploying OSSAF Adapter as an EJB .....	57
3.7.4	Writing an OSSAF Adapter client .....	57
3.8	UCA-EBC Adapter specific configuration.....	58
3.8.1	UCA-EBC Adapter Configuration.....	59
3.8.2	UCA-EBC value pack configurations .....	61
3.8.3	Configuring a value pack for collecting events from an UMB flow .....	61
3.8.4	Forwarding Alarms to UMB through Static flows.....	62
3.8.5	Forwarding Alarms to UMB through Dynamic flows .....	63
<b>Chapter 4</b>	<b>.....</b>	<b>64</b>
<b>Unified Mediation Bus Adapter Development Kit</b>	<b>.....</b>	<b>64</b>
4.1	Licensing .....	64
4.2	Disk requirements.....	64
4.3	Software prerequisites .....	64
4.3.1	Java.....	64
4.4	Unified Mediation Bus Adapter Development Kit installation .....	66
4.4.1	Product Installation.....	66
4.4.2	Files organization .....	68
4.4.3	Setting the Unified Mediation Bus Adapter Development Toolkit environment variables (Linux only) .....	69
4.5	Un-installation of UMB Adapter Development Kit.....	69
<b>Chapter 5</b>	<b>.....</b>	<b>70</b>
<b>Code Signing</b>	<b>.....</b>	<b>70</b>
5.1	On Red Hat Enterprise Linux and HP-UX platforms.....	70
<b>Glossary</b>	<b>.....</b>	<b>71</b>

# Figures

Figure 1 - Unified Mediation Bus architecture overview.....	10
Figure 2 - Example of AdapterConfiguration.xml file.....	36
Figure 3 - Example of adapter.properties file .....	41
Figure 4 - Example of hazelcast.xml file .....	43
Figure 5 - TeMIP adapter overview .....	45
<b>Figure 6 - The TeMIP Adapter's AdapterConfiguration.xml file .....</b>	<b>47</b>
Figure 7 - Executing an Alarm Object directive action on TeMIP Adapter .....	49
Figure 8 - Executing a Trouble Ticket directive action on TeMIP Adapter .....	49
Figure 9 - Executing a Passthrough action on TeMIP Adapter .....	50
Figure 10 - Example of a TeMIP configuration file .....	51
Figure 11 - Example of TeMIP/TWS configuration in the TeMIP_configuration.dynamic.xml file.....	52
Figure 12 - How to query the TeMIP director entity name .....	52
Figure 13 - How to specify the Operation Context(s) in the TeMIP configuration file .....	52
Figure 14 - Adding a custom AO attribute in the TeMIP_configuration.dynamic.xml file.....	53
<b>Figure 15 - Example of an axis2.xml configuration file .....</b>	<b>54</b>
Figure 16 - UCA-EBC adapter architecture.....	59
Figure 17 - Setting the JAVA_HOME environment variable on Windows systems .....	65
Figure 18 - Installing UMB Adapter Development Kit .....	66

## Tables

Table 1 - Software versions .....	7
Table 2 - Software Prerequisites for UMB Server V1.0 components .....	12
Table 3 - Disk Requirements for UMB Server zookeeper component .....	13
Table 4 – ZooKeeper binary Files organization .....	15
Table 5 – ZooKeeper data files organization .....	16
Table 6 – Kafka binary Files organization.....	22
Table 7 – Kafka data files organization .....	22
Table 8 - Software Prerequisites for UMB Adapters.....	28
Table 9 - Disk Requirements for one UMB Runtime .....	31
Table 10 - Sub-directories of UMB Adapter installation directory .....	31
Table 11 - Sub-directories of UMB Adapter installation directory .....	34
Table 12 - Disk Requirements for UMB Development kit .....	64
Table 13 - Software Prerequisites for UMB Adapter Development Kit .....	65
Table 14 - Sub-directories of UMB Adapter Development Kit installation directory .....	68

# Preface

This guide describes how to install the product on the various supported platforms.

Product Name: Unified Correlation Analyzer Mediation

Product Version: 1.0

Kit Version: 1.0

## Intended Audience

Here are some recommendations based on possible reader profiles:

- Solution Developers
- Software Development Engineers

## Software Versions

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

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

Product Version	Supported Operating systems
UMB Server Version V1.0	• Red Hat Enterprise Linux Server release 6.5
UMB Adapters Version V1.0	• HP-UX 11.31 for Itanium • Red Hat Enterprise Linux Server release 6.5 • Windows XP / Vista 64 bits • Windows Server 2012 • Windows 7 64 bits

**Table 1 - Software versions**

## Typographical Conventions

*Courier* Font:

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

*Italic* Text:

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

**Bold Text:**

- To introduce new terms and to emphasize important words.

## Associated Documents

The following documents contain useful reference information:

### References

[R1] *Unified Mediation Bus - Adapter Development Guide*

## Support

Please visit our HP Software Support Online Web site at [www.hp.com/go/hpsoftwaresupport](http://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.
- Patches and updates.
- Problem reporting.
- Training information.
- Support program information.



# Chapter 1

## Introduction

This guide describes the installation and configuration procedures for the Unified Mediation Bus Server, Adapters and Adapter Development Kit products.

## Overview

Unified Mediation Bus allows several applications to exchange Events (and by extension Alarms) with each other. It also provides facilities for executing actions remotely: alarm operations (creation, grouping, deletion etc...), Trouble ticket operations, command executions (shell scripts, java, etc...)

The Unified Mediation Bus product comes in replacement of the legacy “NGOSS Open Mediation” product with the aim to provide:

- Better performance
- Better robustness
- Easier deployment
- Easier Adapter Development

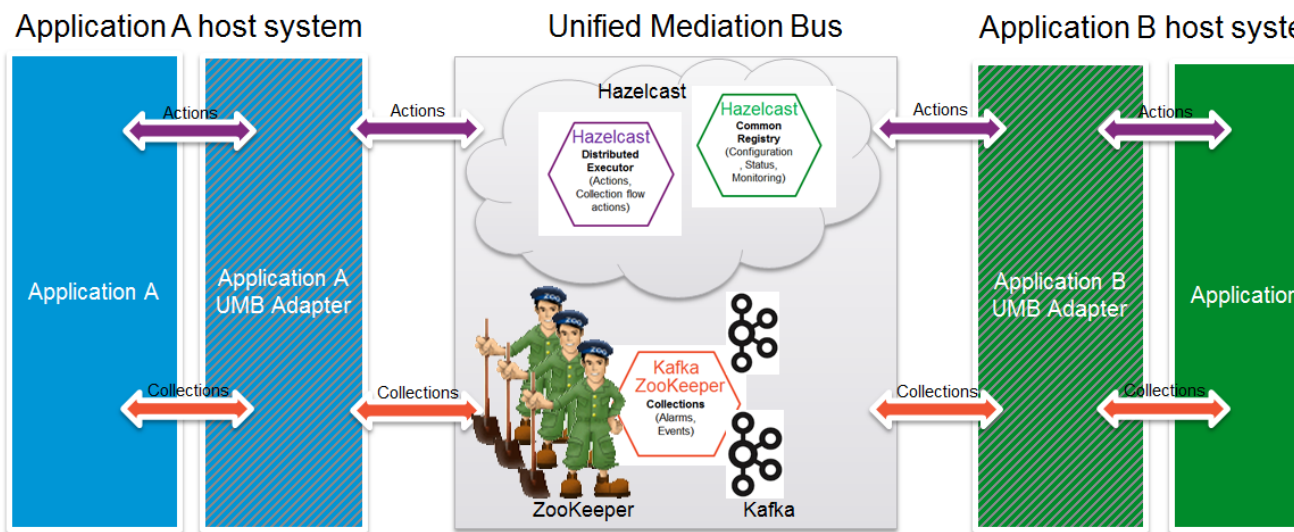
Unified Mediation Bus is constructed around two main technologies:

- A common registry, and remote execution service implemented with the Hazelcast® technology. Hazelcast provides both:
  - a common registry feature that centralizes configuration, status and monitoring information on all UMB Adapters that are part of the overall UMB solution
  - a distributed executor service feature that provides a framework for executing actions on UMB Adapters across the whole UMB solution
- A message broker based on the Kafka Technology. Apache Kafka / Apache ZooKeeper provide a high-performance, high-availability, reliable framework for producing and consuming collections of alarms or events across the whole UMB solution

A typical UMB solution is composed of (see figure below):

- A UMB Server product installation, usually installed on 1 or more dedicated UMB Server host(s), that contains Apache Kafka / Apache ZooKeeper

- Several UMB Adapter<sup>1</sup> product installations (one for each Application connected to the UMB solution). Each application has its own dedicated UMB Adapter, usually installed on the same host as the application itself.



**Figure 1 - Unified Mediation Bus architecture overview**

The above figure shows UMB interconnecting 2 separate applications: Application A and Application B.

In the figure, Hazelcast appears as a centralized component for simplification's sake: Hazelcast is in fact distributed across both Application A and Application B UMB Adapters. Each of the UMB Adapters is a Hazelcast cluster member. Hazelcast cluster members are interconnected directly, without any centralized component. Any UMB Adapter can act as an action service provider and/or consumer:

- It provides action services for the Application that it is associated with (in our case Application A or Application B). UMB Adapters act as proxies to execute actions on Applications that they are associated with.
- It consumes action services from other UMB Adapters

On the other hand, Apache Kafka / Apache ZooKeeper are indeed a centralized component. Both Application A and Application B UMB Adapters connect to the same central component. Apache ZooKeeper provides a high performance coordination service for the "cluster" of Apache Kafka brokers. Apache ZooKeeper acts as a front-end to the Apache Kafka brokers. The Apache Kafka brokers provide the messaging service: they store collections of alarms or events (sent by Kafka producers) as Topics. Kafka consumers then retrieve the collections of alarms or events. Any UMB Adapter can act as Kafka producer and/or Kafka consumer:

- It provides collection services for the Application that it is associated with (in our case Application A or Application B). UMB Adapters act as proxies to collect alarms or events from Applications that they are associated with.
- It consumes collection services from other UMB Adapters

<sup>1</sup> UMB Adapters are developed using the UMB Adapter Development Kit. Information on how to install the UMB Adapter Development Kit is provided in chapter: Chapter 4

# Unified Mediation Bus Server

The UMB Server product is only available on Linux. The installation procedures described here below therefore only apply to the Linux System.

The Unified Mediation Bus Server is based on the Apache Kafka distributed messaging system. For Kafka to work properly, it requires the installation of the Apache ZooKeeper application which will be in charge of enabling the reliability and coordination between the different Kafka servers.

The UMB Server product is therefore made of two different packages:

- umb-zookeeper-package-1.0-linux.tar
- umb-kafka-package-1.0-linux.tar

This chapter describes the software prerequisites, installation steps, and gives a brief content description of the UMB Server kits.

## 2.1 Licensing

No extra license is required to run a Unified Mediation Bus Server.

## 2.2 UMB Server configurations overview

The UMB Server, through the use of Zookeeper and Kafka offers different level of availability depending on the way it is configured.

### **Standalone Configuration:**

The Standalone configuration is the simplest configuration for the UMB Server. With this configuration, both 'zookeeper' and 'kafka' are installed on the same server. These services are not replicated and as a consequence stopping one of these two services will interrupt the overall Mediation functionality.

The configuration is suitable for demonstrations but is not recommended for a production environment.

### **Highly available Configuration:**

With this configuration both the zookeeper and kafka services are duplicated on different servers.

By setting this configuration, the solution is protected against the crash of one of the server and therefore is made highly available.

The configuration is recommended for production environment.

Refer to section 2.6 "Configuring UMB server for production environment" for details on how to configure the UMB server for a production environment.

## 2.3 Pre-installation tasks

### 2.3.1 Java

Both Unified Mediation Bus Server V1.0 zookeeper and kafka components requires the installation of a Java JRE as prerequisite.

Software	Version
Java JRE/JDK 7	1.7.0.00 (or later)

**Table 2 - Software Prerequisites for UMB Server V1.0 components**

To check if you already have Java installed:

```
$ rpm -qa | grep jdk
```

Red Hat Enterprise Linux Server comes with OpenJDK Java VM. You should get an output similar to the following (if 1.7.0 is installed):

```
java-1.7.0-openjdk-1.7.0.9-2.3.4.1.el6_3.x86_64
java-1.7.0-openjdk-devel-1.7.0.9-2.3.4.1.el6_3.x86_64
```

You can also download (for free) the latest Java packages (HotSpot Java VM) from Oracle from <http://java.com/en/download/manual.jsp>. If this is installed (usually under `/usr/java`), you should get an output similar to the following:

```
jdk-1.7.0_51-fcs
```

### 2.3.2 Create UMB administration User 'hpossadm'

#### Notes

- This step is not mandatory if you install UMB Server V1.0 components as a non-root user. In such case the user that you use for installing the packages will become the administration user for UMB.
- When installing as root, an 'hpossadm' user must be created on the system, you can creat it with the following instructions, or ask your system administrator to create it according to your compagny policies. If the 'hpossadm' user is not created before UMB packages installation, the installation script will create it for you, using basic settings.

The local "hpossadm" user account must have a `${HOME}` directory containing at least a `.login` or a `.profile` file.

The following super user command should create an acceptable "hpossadm" user:

```
$ useradd -g <your hpossadm group name here> -m -d
/home/hpossadm -s /bin/bash hpossadm
```

## 2.4 UMB zookeeper installation

### 2.4.1 Disk requirements

Here are the disk requirements for UMB Server zookeeper component:

Type	Disk requirements
Installation time temporary disk space	20 MB minimum: <ul style="list-style-type: none"><li>• 10 MB minimum for the <i>umb-zookeeper-package-1.0-linux.tar</i> file</li><li>• 10 MB minimum for installation files (expanded from the <i>umb-zookeeper-package-1.0-linux.tar</i> file)</li></ul>
Permanent disk space	10 MB minimum for UMB Server V1.0 installed on the system

**Table 3 - Disk Requirements for UMB Server zookeeper component**

### 2.4.2 UMB Server zookeeper package installation

The current version of Zookeeper used by UMB Framework is 3.4.6.

#### 2.4.2.1 Untar the archive in a temporary directory

Untar the archive in a temporary local directory (For example: /tmp):

```
$ cd /tmp
$ tar -xvf <kit location>/umb-zookeeper-package-1.0-linux.tar
```

The output should be as follows:

```
UMBzookeeper-V1.0-0A.noarch.rpm
install-umb-zookeeper.sh
$
```

#### 2.4.2.2 Run the installation script

As root user, run the package installation script:

```
$ install-umb-zookeeper.sh
```

This command installs the package by default in the `/opt/UMB/zookeeper` directory and stores the zookeeper data part in the `/var/opt/UMB/zookeeper` directory.

The following options can be specified for changing these default values:

**-r root\_directory** : Specifies a valid Unified Mediation Bus zookeeper Root Directory (default=`/opt/UMB/zookeeper`)

**-d data\_directory** : Specifies a valid Unified Mediation Bus zookeeper Data Directory (default=`/var/opt/UMB/zookeeper`)

---

**Note**

---

When installed as root user the following is performed during installation

- A 'hpossadm' user is automatically created if not already there on the system. In case of automatic creation a password must be assigned to the created user in order to be able to log in. As root user, use the '`passwd hpossadm`' command to change the password.
  - A linux 'zookeeper' service is created in order to ease the monitoring and offer the possibility to automatically start the service on boot.
- 

### Installing UMB Server ZooKeeper package as non-root user:

For testing purposes (or for some very specific needs) the UMB Server ZooKeeper package can be installed by a non-root user (Note that this is not the recommended way for installing this package).

When installing UMB Server ZooKeeper package as non-root user, the following limitations must be understood and acknowledged:

- The system RPM database is not accessible by a non-root user. As a consequence, when installation is performed by a non-root user, a specific RPM database must be specified. The default RPM repository for non-root installation is set to `~/rpmdb` (where `~` is the user home directory). This directory can be overridden by specifying the `--rpmdbpath` option as installation script argument.
- The UMB Server ZooKeeper package binary and data directories must be read/write accessible by the non-root user. Usually the default `/opt/UMB/zookeeper` and `/var/opt/UMB/zookeeper` directories cannot be used (unless some specific rights have been set by the administrator). As a consequence, when installation is performed by a non-root user, the `-r` and `-d` options **must** be specified.
- When installed by the non-root users the UMB Server ZooKeeper binaries and scripts will only be executable by the user who did perform the installation. As a consequence UMB Server ZooKeeper administration (start/stop/status) has to be executed with this user (and not as 'uca' user as stated in the documentation)
- Finally the ZooKeeper linux service will not be created. As a consequence it will not be possible to use the "automatic start on boot" as described in section 2.4.2.7

#### 2.4.2.3 Starting and stopping ZooKeeper

From the package installation `bin` directory (default `/opt/UMB/zookeeper/bin`) :

##### Starting Zookeeper:

As 'hpossadm' user (or from the user who did the installation in case of non-root installation)

```
zookeeper start
```

##### Checking ZooKeeper status:

As 'hpossadm' user (or from the user who did the installation in case of non-root installation)

```
zookeeper status
```

#### Stopping ZooKeeper service:

As 'hpossadm' user (or from the user who did the installation in case of non-root installation)

```
zookeeper stop
```

When installed as root user, the UMB Server ZooKeeper component can be started as a Linux service.

#### Starting ZooKeeper service:

As root user

```
/sbin/service zookeeper start
```

Checking zooKeeper service status:

As root user

```
/sbin/service zookeeper status
```

Stopping zooKeeper service:

As root user

```
/sbin/service zookeeper stop
```

### 2.4.2.4 Files organization

The UMB Server zookeeper binary files are installed in the root directory specified at installation (by default /opt/UMB/zookeeper)

The following table describes the different sub-directories of ZooKeeper binary part:

Directories	Description
default	Contains default configuration files
usr/bin	Contains ZooKeeper command-line scripts
usr/sbin	Contains additional ZooKeeper command-line scripts
usr/share/zookeeper	Contains the ZooKeeper libraries

**Table 4 – ZooKeeper binary Files organization**

The following table describes the different sub-directories of ZooKeeper data part (by default /var/opt/UMB/zookeeper):

Directories	Description
config	Contains customized configuration files

Directories	Description
lib/data	Contains the ZooKeeper runtime data
log	Contains ZooKeeper log files (traces)
run	Contains ZooKeeper pid file

**Table 5 – ZooKeeper data files organization**

#### 2.4.2.5 TCP ports used by ZooKeeper

ZooKeeper uses the following TCP ports:

- 2181 by default (see the `clientPort` property in the `zoo.cfg` file located by default in the `/var/opt/UMB/zookeeper/config` folder): used for client connections

Additionally, if ZooKeeper is running in highly available configuration (i.e. there is more than 1 ZooKeeper node in the ensemble), it uses the following extra TCP ports:

- 2888 by default (see the `server.<server number>` property in the `zoo.cfg` file): used for followers/quorum connections
- 3888 by default (see the `server.<server number>` property in the `zoo.cfg` file): used for leader election connections

In the `zoo.cfg` file, the entries of the form `server.<server number>` list the servers that make up the ZooKeeper service. When the server starts up, it knows which server it is by looking for the file `myid` file in the ZooKeeper data directory: `/var/opt/UMB/zookeeper/lib/data/myid`. This file contains the server number, in ASCII.

The format for the `server.<server number>` property is the following:

```
server.<server number>=<server host name>:<followers
port>:<leader election port>
```

For example: `server.1=host1:2888:3888`

The two port numbers after each server name: 2888 and 3888. Peers use the former port to connect to other peers. Such a connection is necessary so that peers can communicate, for example, to agree upon the order of updates. More specifically, a ZooKeeper server uses this port to connect followers to the leader. When a new leader arises, a follower opens a TCP connection to the leader using this port. Because the default leader election also uses TCP, we currently require another port for leader election. This is the second port in the server entry.

#### Note

In the case of multiple servers on a single machine (this configuration is not recommended in a production environment), please specify the `server host name` as `localhost` with unique quorum & leader election ports (i.e. `2888:3888`, `2889:3889`, `2890:3890` in the example above) for each `server.<server number>` in that server's configuration file. Of course separate `dataDir` and distinct `clientPort` values are also necessary (in the above replicated example, running on a single localhost, you would still have three configuration files).



---

### 2.4.2.6 Red Hat Linux firewall settings

As detailed above, UMB Server ZooKeeper functionality uses some ports that need to be open on the firewall in order for ZooKeeper and Kafka to run properly. Please see chapter 2.4.2.5 “TCP ports used by ZooKeeper” for more information on the ports<sup>2</sup> used by ZooKeeper.

Let’s suppose we have the default `iptables` configuration file, like the following:

```
# cat /etc/sysconfig/iptables
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j
ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

You will need to add filters to your current `iptables` settings to open ports used by each ZooKeeper instance that are part of your UMB Server setup.

By default, 3 chains are used: INPUT, OUTPUT, FORWARD. Please refer to the Red Hat Linux guide for a better understanding of what a chain is and what the packet matching rules are.

Here we are going to create a new custom INPUT chain, dedicated to managing UMB ZooKeeper ports.

Let’s call it UMB. To do so, you will need to:

- Add 2 lines to define the UMB chain:

```
:UMBZookeeper - [0:0]
-A INPUT -j UMBZookeeper
```

- Add 1 line for each ZooKeeper instance used in your UMB Server setup:

```
-A UMBZookeeper -p tcp -m multiport --dports 2181,2888,3888
-m comment --comment "UMB ZooKeeper instance 1" -j ACCEPT
```

---

<sup>2</sup> All ports used by ZooKeeper are TCP ports

Please make sure to use the same port numbers as the ones defined in both the ZooKeeper `zoo.conf` file for every ZooKeeper instance and the Kafka `server.properties` file for every Kafka Broker.

Please see below for an updated version of the configuration file (added lines are in [blue](#)):

```
# cat /etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2843:756710]
: UMBZookeeper - [0:0]
-A INPUT -j UMBZookeeper
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j
ACCEPT
-A UMBZookeeper -p tcp -m multiport --dports 2181,2888,3888 -m
comment --comment "UMB ZooKeeper instance 1" -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Then, you need to validate your settings using the `iptables` command or the `iptables` service.

```
# service iptables restart
iptables: Flushing firewall rules:           [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules:                 [ OK ]
iptables: Applying firewall rules:           [ OK ]
```

And then you need to check that your new UMBZookeeper settings are up and running:

```
# iptables --list UMBZookeeper
Chain UMB (1 references)
target      prot opt source                destination
ACCEPT      tcp  --  anywhere              anywhere
multiport dports 2181,2888,3888 /* UMB ZooKeeper instance 1 */
```

Please make sure to open the ports for all ZooKeeper instances and Kafka brokers running on your server.

If everything is OK, please save your configuration so that it is taken into account after a reboot:

```
# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[
OK ]
```

#### 2.4.2.7 Auto-starting Zookeeper service on boot

When installed as root user, the UMB Server ZooKeeper package also installs the 'zookeeper' service. As with any other services, the zookeeper service can be auto-started on boot.

- Become as root user on your Linux server

- Add script to start on boot using chkconfig utility:

```
$ chkconfig --add zookeeper
$ chkconfig zookeeper on
```

- Confirm script is added using chkconfig utility

```
$ chkconfig --list zookeeper
zookeeper 0:off 1:off 2:on 3:on 4:on 5:on 6:off
$
```

## 2.4.3 Un-installation of UMB Server ZooKeeper package

In order to un-install UMB Server ZooKeeper package, you first need to stop ZooKeeper service:

Stop ZooKeeper:

```
zookeeper stop
```

or stop zookeeper service when zookeeper is started as a service:

```
service zookeeper stop
```

Once ZooKeeper service is stopped, you then need to un-install the UMB Server ZooKeeper rpm packages:

```
$ rpm -e UMBzookeeper-V1.0-0A
```

## 2.5 UMB Kafka installation

### 2.5.1 UMB Server kafka package installation

The current version of Kafka used by UMB Framework is 0.8.2.0.

#### 2.5.1.1 Untar the archive in a temporary directory

Untar the archive in a temporary local directory (For example: /tmp):

```
$ cd /tmp
$ tar -xvf <kit location>/umb-kafka-package-1.0-linux.tar
```

The output should be as follows:

```
UMBkafka-V1.0-0A.noarch.rpm
install-umb-kafka.sh
$
```

### 2.5.1.2 Run the installation script

As root user, run the package installation script:

```
$ install-umb-kafka.sh
```

This command installs the package by default in the `/opt/UMB/kafka` directory and stores the kafka data part in the `/var/opt/UMB/kafka` directory.

The following options can be specified for changing these default values:

**-r root\_directory** : Specifies a valid Unified Mediation Bus kafka Root Directory (default=`/opt/UMB/kafka`)

**-d data\_directory** : Specifies a valid Unified Mediation Bus kafka Data Directory (default=`/var/opt/UMB/kafka`)

---

#### Note

When installed a root user the following is performed during installation

- A 'hpossadm' user is automatically created if not already there on the system. In case of automatic creation a password must be assigned to the created user in order to be able to log in. As root user, use the 'passwd hpossadm' command to change the password.
- a linux 'kafka' service is created in order to ease the monitoring and offer the possibility to automatically start the service on boot.

---

### Installing UMB Server kafka package as non-root user:

For testing purpose (or for some very specific needs) the UMB Server kafka package can be installed by a non-root user (Note that this is not the recommended way for installing this package).

When installing UMB Server kafka package as non-root user, the following limitations must be understood and acknowledged:

- The system RPM database is not accessible by a non-root user. As a consequence, when installation is performed by a non-root user, a specific RPM database must be specified. The default RPM repository for non-root installation is set to `~/rpmdb` (where `~` is the user home directory). This directory can be overridden by specifying the `--rpmdbpath` option as installation script argument.
- The UMB Server kafka package binary and data directories must be read/write accessible by the non-root user. Usually the default `/opt/UMB/kafka` and `/var/opt/UMB/kafka` directories cannot be used (unless some specific rights have been set by the administrator). As a consequence, when installation is performed by a non-root user, the `-r` and `-d` options **must** be specified.
- When installed by the non-root users the UMB Server kafka binaries and scripts will only be executable by the user who did perform the installation. As a consequence UMB Server kafka administration (start/stop/status) has to be executed with this user (and not as 'uca' user as stated in the documentation)
- Finally the kafka linux service will not be created. As a consequence it will not be possible to use the "automatic start on boot" as described in section 2.5.1.7

### 2.5.1.3 Starting and stopping kafka

From the package installation `bin` directory (default `/opt/UMB/kafka/bin`) :

#### Starting kafka:

As 'hpossadm' user (or from the user who did the installation in case of non-root installation)

```
kafka start
```

#### Checking kafka status:

As 'hpossadm' user (or from the user who did the installation in case of non-root installation)

```
kafka status
```

#### Stopping kafka service:

As 'hpossadm' user (or from the user who did the installation in case of non-root installation)

```
kafka stop
```

When installed as root user, the UMB Server kafka component can be started as a Linux service.

---

#### Note:

Before applying the here below commands, the following modification must be performed:

Edit the file `/etc/rc.d/init.c/kafka`

Add the following line at the top of the file :

```
# chkconfig: 2345 89 9
```

The file header should look like:

```
#  
# Kafka  
#  
# chkconfig: 2345 89 9  
# description: kafka
```

---

#### Starting kafka service:

As root user

```
/sbin/service kafka start
```

Checking kafka service status:

As root user

```
/sbin/service kafka status
```

Stopping kafka service:

As root user

```
/sbin/service kafka stop
```

#### 2.5.1.4 File organization

The UMB Server kafka binary files are installed in the root directory specified at installation (by default `/opt/UMB/kafka`)

The following table describes the different sub-directories of Kafka binary part:

Directories	Description
<i>bin</i>	Contains the Kafka shell scripts
<i>defaults</i>	Contains default configuration files
<i>libs</i>	Contains the Kafka libraries
<i>logs</i>	Contains the Kafka log files

**Table 6 – Kafka binary Files organization**

The following table describes the different sub-directories of Kafka data part (by default `/var/opt/UMB/kafka`):

Directories	Description
<i>config</i>	Contains customized configuration files
<i>kafka-logs</i>	Contains the Kafka data files
<i>logs</i>	Contains Kafka log files (traces)
<i>run</i>	Contains Kafka pid file

**Table 7 – Kafka data files organization**

#### 2.5.1.5 TCP ports used by Kafka

Kafka uses the following TCP ports:

- 9092 by default (see the `port` property in the `server.properties` file located by default in the `/var/opt/UMB/kafka/config` folder): used by the Kafka broker

In case of multiple-broker configuration, each Kafka broker has its own `server.properties` configuration file that defines the `port` property that contains the port number used by the broker.

#### Note

In the case of multiple Kafka brokers on a single machine (this configuration is not recommended in a production environment), please specify different value for the `port` property for each Kafka broker. In this case, it is customary to just increase the port number by 1 for each additional Kafka broker. For example: port 9093 for Kafka broker 2, 9094 for Kafka broker 3, etc...

### 2.5.1.6 Red Hat Linux firewall settings

UMB Server kafka component uses some ports that need to be open on the firewall in order for ZooKeeper and Kafka to run properly. Please see chapter 2.5.1.5 “TCP ports used by Kafka” for more information on the ports<sup>3</sup> used by Kafka.

Let’s suppose we have the default `iptables` configuration file, like the following:

```
# cat /etc/sysconfig/iptables
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j
ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

You will need to add filters to your current `iptables` settings to open ports used by each ZooKeeper instance and each Kafka broker that are part of your UMB Server setup.

By default, 3 chains are used: INPUT, OUTPUT, FORWARD. Please refer to the Red Hat Linux guide for a better understanding of what a chain is and what the packet matching rules are that apply within a chain.

Here we are going to create a new custom INPUT chain, dedicated to managing UMB ports.

Let’s call it UMBKafka. To do so, you will need to:

- Add 2 lines to define the UMBKafka chain:

```
:UMBKafka - [0:0]
-A INPUT -j UMBKafka
```

- Add 1 line for the Kafka broker:

```
-A UMBKafka -p tcp -m tcp --dport 9092 -m comment --comment
"UMB Kafka broker 1" -j ACCEPT
```

Please make sure to use the same port number as the ones defined in the Kafka `server.properties` file.

---

<sup>3</sup> All ports used by Kafka are TCP ports

Please see below for an updated version of the configuration file (added lines are in blue):

```
# cat /etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2843:756710]
:UMBKafka - [0:0]
-A INPUT -j UMBKafka
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j
ACCEPT
-A UMBKafka -p tcp -m tcp --dport 9092 -m comment --comment
"UMB Kafka broker 1" -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Then, you need to validate your settings using the `iptables` command or the `iptables` service.

```
# service iptables restart
iptables: Flushing firewall rules:          [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules:                [ OK ]
iptables: Applying firewall rules:          [ OK ]
```

And then you need to check that your new UMBKafka settings are up and running:

```
# iptables --list UMBKafka
Chain UMBKafka (1 references)
target      prot opt source                destination
ACCEPT      tcp  --  anywhere               anywhere
tcp dpt:9092 /* UMB Kafka broker 1 */
```

If everything is OK, please save your configuration so that it is taken into account after a reboot:

```
# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[
OK ]
```

### 2.5.1.7 Auto-starting kafka service on boot

When installed has root user, the UMB Server kafka package also install the 'kafka' service. As any other services, the kafka service can be auto-started on boot.

- Become as root user on your Linux server
- Add script to start on boot using `chkconfig` utility:

```
$ chkconfig --add kafka
$ chkconfig kafka on
```

- Confirm script is added using `chkconfig` utility

```
$ chkconfig --list kafka
kafka 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```



```
$
```

## 2.5.2 Un-installation of UMB Server Kafka package

In order to un-install UMB Server Kafka package, you first need to stop Kafka service:

Stop Kafka:

```
service kafka stop
```

Once Kafka service is stopped, you then need to un-install the UMB Server Kafka rpm packages:

```
$ rpm -e UMBkafka-V1.0-0A
```

## 2.6 Configuring UMB server for production environment

Production environments usually require Fault Tolerance mechanisms to be put in place in order to prevent service disruption in case of a system crash.

For the UMB server this means two things:

1. Implement a ZooKeeper Cluster
2. Make the Kafka broker redundant

### 2.6.1 Configuring ZooKeeper in a Production Environment

For use in a production environment, ZooKeeper should be deployed as an ensemble with an odd number of nodes. As long as a majority of the servers in the ensemble are available, the ZooKeeper service will be available. The minimum recommended ensemble size is three ZooKeeper servers, and it is recommended that each server run on a separate machine.

It is recommended to run at least three ZooKeeper nodes on three separate machines to meet the high availability requirements. If there are sufficient resources, it is recommended to run five ZooKeeper nodes, which allows one node to be taken down for planned maintenance without affecting the availability.

ZooKeeper deployment on multiple servers requires a bit of additional configuration. The configuration file (`zoo.cfg`) on each server must include a list of all servers in the ensemble, and each server must also have a `myid` file in its data directory (by default: `/var/opt/UMB/zookeeper/lib/data`) that identifies it as one of the servers in the ensemble.

The ids of the ZooKeeper servers must be unique in the ensemble. Each ZooKeeper is to be installed on a separate server.

ZooKeeper's behavior is governed by the ZooKeeper configuration file. This file is designed so that the exact same file can be used by all the servers that make up a ZooKeeper ensemble assuming the disk layouts are the same. If servers use different configuration files, care must be taken to ensure that the list of servers in all of the different configuration files match.

## Assign each ZooKeeper instance an Identifier

Each ZooKeeper server is identified by an identifier. This identifier is stored in the file 'myid' in the ZooKeeper's data directory (default:

`/var/opt/UMB/zookeeper/lib/data.`

You can assign this identifier by means of an `init` command option of the `zookeeper` shell script:

```
$ zookeeper init 1
```

This will give this ZooKeeper instance the identifier '1'.

## Configure the servers in order they know each other's

For the different servers part of the ZooKeeper cluster to be able to cooperate they must know each other's.

This is done by adding neighborhood references to each server configuration in the server configuration file `zoo.cfg`.

Imagine that the ZooKeeper cluster is made of three members running on `host1`, `host2` and `host3`

The zookeeper identifier given to each member is respectively 1, 2 and 3.

The configuration file of each member must then be augmented with the following properties:

```
server.1=host1:2888:3888
server.2=host2:2888:3888
server.3=host3:2888:3888
```

## Change the Kafka server configuration in order for it to use the Zookeeper cluster configuration

The Kafka broker is aware of the Zookeeper configuration through the `zookeeper.connect` property which defines the zookeeper connection string.

In case of a Zookeeper Cluster configuration, the Connection String is defined with a coma separated list of Zookeeper URLs as follow:

```
# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify
# the
# root directory for all kafka znodes.
zookeeper.connect=host1:2181,host2:2181,host3:2181
```

## Change the adapter's configurations in order for them to use the Zookeeper cluster configuration

Each adapter consuming events must define a Zookeeper connection String through the property `consumer.zookeeper.connect` in the `adapter.properties` file.

In case of a Zookeeper Cluster configuration, the Connection String is defined with a coma separated list of Zookeeper URLs as follow:

```
consumer.zookeeper.connect=host1:2181,host2:2181,host3:2181
```

## 2.6.2 Make the Kafka Broker Redundant

In order to secure the Kafka Broker's data, such data must be replicated.

This is done by defining two or more Kafka Brokers on which the UMB topics are replicated. The servers must be located on different systems.

On each of these brokers the Kafka configuration file (default: `/var/opt/UMB/kafka/config/server.properties`) must be changed as follow:

### Assign each Kafka instance a unique Identifier

This is done by setting the `broker.id` property to a unique value

Example:

On the first Kafka broker

```
# The id of the broker. This must be set to a unique integer
for each broker.
broker.id=1
```

On the second Kafka broker

```
# The id of the broker. This must be set to a unique integer
for each broker.
broker.id=2
```

### Set the Default replication factor

On each broker sets the `default.replication.factor` property to a value of 2 (when two brokers are configured) or 3 if more than two brokers are configured.

```
default.replication.factor=2
```

### Set the Zookeeper connection String

On each broker sets the `zookeeper.connect` property to a value similar to the one from the ZooKeeper Cluster setting

Example:

```
zookeeper.connect=host1:2191,host2:2181,host3:2181
```

### Change the adapter's configurations in order for them to use the Kafka cluster configuration

Each adapter producing events must define a broker connection String through the property `producer.metadata.broker.list` in the `adapter.property` file.

In case of a Kafka redundancy configuration, the broker list is defined with a coma separated list of Kafka URLs as follow:

Example:

```
producer.metadata.broker.list=host1:9092,host2:9092
```

# Unified Mediation Bus Adapters

Unified Mediation Bus Adapters are the intermediate processes that insure connectivity and data conversion between applications connected through the Unified Mediation Bus. A Unified Mediation Bus Adapter can be separate from (external) or embedded into the 3<sup>rd</sup> party application.

When embedded, the 3<sup>rd</sup> party application is itself an adapter (this is the case for UCA-EBC for example).

When external, the adapter is delivered as a separate installable package (examples: Exec Adapter, TeMIP Adapter, File Adapter, Camel Adapter, etc...

A Unified Mediation Bus Adapter can be installed both on HP-UX or Linux systems.

This chapter gives details on how to install and configure Unified Mediation Bus Adapters.

## 3.1 Licensing

No extra license is required to run the Unified Mediation Bus Adapters.

### Note

Please refer to Chapter 2.1 “*Licensing*” for more information on UMB licensing.

## 3.2 Installation pre-requisites

### 3.2.1 Java

Unified Mediation Bus V1.0 Adapter requires Java 1.7.

Software	Version
Java JRE/JDK 7	1.7.0.00 (or later)

**Table 8 - Software Prerequisites for UMB Adapters**

#### On HP-UX:

To check if you already have Java installed:

```
$ swlist | grep Java
```

You should get an output similar to the following:

Java70JRE	1.7.0.10.00	Java 7.0 JRE for HP-UX
-----------	-------------	------------------------

The latest JDK package for HP-UX can be downloaded (for free) from [www.hp.com/go/java](http://www.hp.com/go/java).

It is usually installed in the `/opt/java7` folder.

#### On Linux:

To check if you already have Java installed:

```
$ rpm -qa | grep jdk
```

Red Hat Enterprise Linux Server comes with OpenJDK Java VM. You should get an output similar to the following (if 1.7.0 is installed):

```
java-1.7.0-openjdk-1.7.0.9-2.3.4.1.el6_3.x86_64
java-1.7.0-openjdk-devel-1.7.0.9-2.3.4.1.el6_3.x86_64
```

You can also download (for free) the latest Java packages (HotSpot Java VM) from Oracle from <http://java.com/en/download/manual.jsp>. If this is installed (usually under `/usr/java`), you should get an output similar to the following:

```
jdk-1.7.0_51-fcs
```

## 3.2.2 Admin user creation

Before installing a UMB Adapter on a system, you need to create a local "hpossadm" user account on that system.

The local "hpossadm" user account must have a `$(HOME)` directory containing at least a `.login` or a `.profile` file (in order to set the UMB Adapter environment variables inside either of these files).<sup>4</sup>

The following super user command should create an acceptable "hpossadm" user:

#### On HP-UX:

```
$ useradd -g <your hpossadm group name here> -m -d
/home/hpossadm -s /bin/csh hpossadm
```

#### On Linux:

```
$ useradd -g <your hpossadm group name here> -m -d
/home/hpossadm -s /bin/bash hpossadm
```

## 3.3 Unified Mediation Bus Runtime Package installation

### 3.3.1 Product installation

Before installing a UMB Adapter on a system, you need to install the UMB Runtime package used by all adapters. The installed package contains all common libraries and UMB framework library.

---

<sup>4</sup> Please see chapter 3.4.5 "Setting the Unified Mediation Bus Adapter environment variables (Unix only)" for more information on how to set these variables.

The installation procedure is as described below.

### 3.3.1.1 Un-tar the archive in a temporary directory

As root user, un-tar the UMB Runtime archive file under a temporary local directory (For example: /tmp):

**On HP-UX:**

```
$ cd /tmp
$ tar -xvf <kit location>/umb-runtime-package-1.0-hpux.tar
```

**On Linux:**

```
$ cd /tmp
$ tar -xvf <kit location>/umb-runtime-package-1.0-linux.tar
```

### 3.3.1.2 Run the installation script

Still as root user, run the package installation script:

**On both HP-UX and Linux:**

```
$ install-umb-runtime.sh
```

This command installs the package by default in the /opt/UMB.

The following options can be specified for changing this default value:

**-r root\_directory** : Specifies a valid Unified Mediation Bus Root Directory (default=/opt/UMB)

---

#### Note

##### Installing a UMB Runtime as non-root user (Linux only):

For testing purpose (or for some very specific needs) the UMB Runtime package can be installed by a non-root user. This feature is available for Linux only.

When installing a UMB Runtime as non-root user, the following limitations must be understood and acknowledged:

1. The system RPM database is not accessible by a non-root user. As a consequence, when installation is performed by a non-root user, a specific RPM database must be specified. The default RPM repository for non-root installation is set to ~/ .rpmdb (where ~ is the user home directory). This directory can be overridden by specifying the --rpmdbpath option as installation script argument.
  2. The UMB root directory must be read/write accessible by the non-root user. As a consequence, when installation is performed by a non-root user, the -r and -o options **must be** specified. More over this user must be the same as the one who installed the OSS Open Mediation packages.
  3. When installed by a non-root user the UMB Runtime files are owned by the user who performed the installation.
- 

## 3.3.2 Disk requirements for the UMB runtime

Here are the minimum disk requirements for each UMB Runtime kit:

Type	Disk requirements
Installation time temporary disk space	44 MB minimum in a temporary directory: <ul style="list-style-type: none"> <li>• 22 MB minimum for the delivered tar file</li> <li>• 22 MB minimum for the extracted files (installer and package)</li> </ul>
Permanent disk space	<ul style="list-style-type: none"> <li>• 25 MB minimum in the installation directory (by default <code>/opt/UMB</code>)</li> </ul>

**Table 9 - Disk Requirements for one UMB Runtime**

### 3.3.3 Files organization

The UMB Runtime is installed in the root directory specified at installation (by default `/opt/UMB`)

The following table describes the different sub-directories contained in the delivery:

Directory	Description
<code>&lt;Adaptername&gt;/bin</code>	Contains the UMB runtime scripts
<code>&lt;Adaptername&gt;/license</code>	Contains the UMB runtime license file.
<code>&lt;Adaptername&gt;/lib</code>	Contains the UMB Framework libraries

**Table 10 - Sub-directories of UMB Adapter installation directory**

### 3.3.4 Setting the Unified Mediation Bus Runtime environment variable (Unix only)

An environment variable must be defined for the installed UMB Runtime to work properly:

- The `${UMB_RUNTIME_HOME}` environment variable references the root directory of the UMB Runtime (default value = `/opt/UMB`)

Additionally the user's path must be upgraded with the `${UMB_RUNTIME_HOME}/bin` directory

For that purpose, the UMB Runtime installation script installs two files in the UMB HOME directory (`/opt/UMB` by default):

- `.adapter_environment.sh`
- `.adapter_environment.csh`

These files can be used for setting the correct environment variables.

Depending on your shell, use one of the following commands to set the "hpossadm" user's UMB environment variables and update the path:

On **csh**-like shell:

```
$ source /opt/UMB/<adapter name>/.adapter_environment.csh
```

On **sh**-like shell:

```
$ . /opt/UMB/<adapter name>/.adapter_environment.sh
```

It is recommended to execute either of these commands inside the `.login` or `.profile` file of the local “hpossadm” user account so that the UMB Runtime environment is always correctly set for this user.

### 3.3.5 Un-installation of a Unified Mediation Bus Runtime

#### 3.3.5.1 On Linux

```
$ rpm -e UMBRUNTIME-V1.0-0A
```

#### 3.3.5.2 On Windows

1. Go to the **Control Panel**
2. Select **“Program and Features”**
3. Right-click on **“HP Unified Mediation Bus Runtime for Adapters – UMBRUNTIME-V1.0-0A”**
4. Select **“Uninstall”**

## 3.4 Unified Mediation Bus Adapters installation

### 3.4.1 Product installation

The installation procedure is identical for any (HP delivered) UMB Adapter. The Exec Adapter will be taken as an example in the sections below.

#### 3.4.1.1 Un-tar the archive in a temporary directory

As root user, un-tar the UMB Adapter archive file under a temporary local directory (For example: `/tmp`):

**On HP-UX:**

```
$ cd /tmp  
$ tar -xvf <kit location>/umb-exec-adapterpackage-1.0-hpux.tar
```

**On Linux:**

```
$ cd /tmp  
$ tar -xvf <kit location>/umb-exec-adapterpackage-1.0-linux.tar
```



### 3.4.1.2 Run the installation script

Still as root user, run the package installation script:

**On both HP-UX and Linux:**

```
$ install-<Adaptername>-adapter.sh
```

This command installs the package by default in the `/opt/UMB/<Adaptername>` directory and stores the configuration files in the `/var/opt/UMB/<Adaptername>/conf` directory.

The following options can be specified for changing these default values:

**-r root\_directory** : Specifies a valid Unified Mediation Bus Root Directory (default=`/opt/UMB`)

**-d data\_directory** : Specifies a valid Unified Mediation Bus Data Directory (default=`/var/opt/UMB`)

---

#### Note

##### Installing a UMB Adapter as non-root user (Linux only):

For testing purpose (or for some very specific needs) the UMB Adapter package can be installed by a non-root user. This feature is available for Linux only.

When installing a UMB Adapter as non-root user, the following limitations must be understood and acknowledged:

4. The system RPM database is not accessible by a non-root user. As a consequence, when installation is performed by a non-root user, a specific RPM database must be specified. The default RPM repository for non-root installation is set to `~/ .rpmdb` (where `~` is the user home directory). This directory can be overridden by specifying the `--rpmdbpath` option as installation script argument.
5. The UMB Adapter root directory must be read/write accessible by the non-root user. As a Consequence, when installation is performed by a non-root user, the `-r` and `-o` options **must be** specified. More over this user must be the same as the one who installed the OSS Open Mediation packages.
6. When installed by a non-root user the UMB Adapter files are owned by the user who performed the installation.

### 3.4.2 Disk requirements for an UMB adapter

An UMB adapter package provides only libraries and scripts that are specific to this adapter. This can be very small (Ex: 40 KB for the Exec adapter) or bigger if the adapter depends on some specific libraries for connectivity (Ex: the TeMIP adapter is up to 27 MB).

### 3.4.3 Files organization

The UMB Adapter is installed in the root directory specified at installation (by default `/opt/UMB/<AdapterName>`)

The following table describes the different sub-directories contained in the delivery:

Directory	Description
<code>&lt;Adaptername&gt;/bin</code>	Contains the Adapter start script
<code>&lt;Adaptername&gt;/defaults</code>	Contains the Adapter's default configuration files. These files are copied to the data directory ( <code>/var/opt/UMB/&lt;Adaptername&gt;/conf</code> by default) at installation time.
<code>&lt;Adaptername&gt;/lib</code>	Contains the Adapter's specific libraries

**Table 11 - Sub-directories of UMB Adapter installation directory**

### 3.4.4 Ports used

#### Hazelcast

Hazelcast uses the following TCP ports:

- 5701 by default (see the `<network><port>...</port></network>` section in the `hazelcast.xml` file of the UMB Adapter located by default in the `/var/opt/UMB/<AdapterName>/conf` folder): used for connecting to other Hazelcast cluster members

If the port set in the `hazelcast.xml` file is already in use, then Hazelcast will try to use the next available port: 5702, or 5703, or 5704, ... for example.

### 3.4.5 Setting the Unified Mediation Bus Adapter environment variables (Unix only)

Several environment variables must be defined for the installed UMB Adapter to work properly:

- The `${UMB_<adapter name in upper case>ADAPTER_HOME}` environment variable (for example: `${UMB_FILEADAPTER_HOME}`): references the root directory ("static" part) of the UMB Adapter (default value = `/opt/UMB/<adapter name>-adapter`)
- The `${UMB_<adapter name in upper case>ADAPTER_DATA}` environment variable (for example: `${UMB_FILEADAPTER_DATA}`): references the data directory ("variable" part) of the UMB Adapter (default value = `/var/opt/UMB/<adapter name>-adapter`)

For that purpose, the UMB Adapter installation script installs two files in the UMB HOME directory (`/opt/UMB/<adapter name>` by default):

- `.adapter_environment.sh`
- `.adapter_environment.csh`

These files can be used for setting the correct environment variables.

Depending on your shell, use one of the following commands to set the "hpossadm" user's UMB environment variables and update the path:

On **csh**-like shell:

```
$ source /opt/UMB/<adapter name>/.adapter_environment.csh
```

On **sh**-like shell:

```
$ . /opt/UMB/<adapter name>/.adapter_environment.sh
```

It is recommended to execute either of these commands inside the `.login` or `.profile` file of the local “hpossadm” user account so that the UMB Adapter environment variables are always set.

### 3.4.6 Starting and stopping a Unified Mediation Bus Adapter

Starting an UMB adapter:

```
<adapter name>-adapter-start
```

Once the UMB adapter has started, it can be stopped by using Ctrl-C or by killing the process associated with the UMB adapter.

### 3.4.7 Un-installation of a Unified Mediation Bus Adapter

On Linux:

```
$ rpm -qa | grep ADAPTER
UMBEXECADAPTER-1.0-0B.noarch
UMBTEMIPADAPTER-1.0-0B.noarch
UMBOSSAFADAPTER-1.0-0B.noarch
UMBLOGADAPTER-1.0-0B.noarch
UMBFILEADAPTER-1.0-0B.noarch
```

Choose the adapter you want to uninstall, for example the exec adapter:

```
$ rpm -e UMBEXECADAPTER-1.0-0B.noarch
```

## 3.5 Unified Mediation Bus Adapters configuration

The configuration files of any UMB adapter are located in the `${UMB_<adapter name in upper case>ADAPTER_DATA}/conf` folder (for example: `/var/opt/UMB/file-adapter/conf` for the File adapter).

### 3.5.1 AdapterConfiguration.xml file

The `AdapterConfiguration.xml` file defines the configuration for the UMB adapter.

It defines the following:

- Name, group and version of the UMB adapter
- Flow services that the UMB adapter provides
- Action services that the UMB adapter provides
- Automatic consumers (of flow services): flow services from other UMB adapters that the UMB adapter consumes as soon as the UMB adapter is started

Below is an example of an `AdapterConfiguration.xml` file:



**Figure 2 - Example of AdapterConfiguration.xml file**

Please refer to [R1] *Unified Mediation Bus - Adapter Development Guide*, chapter 3.2 “Customizing the created UMB Adapter project” for more information on:

- Customizing the Adapter name
- Adding producer collection flow services
- Adding action services
- Adding consumer collection flows

### 3.5.2 adapter.properties file

The `adapter.properties` file defines the properties for the UMB adapter. For the most part, these properties define how to interact with the UMB Server, i.e. ZooKeeper and Kafka.

The properties can be divided in 2 groups:

- **Producer properties:** these properties are prefixed with “`producer.`”. These are properties that you would normally find in a Kafka `producer.properties` file (without the “`producer.`” prefix). These properties are used for by the producer collection flow services of the UMB adapter.
- **Consumer properties:** these properties are prefixed with “`consumer.`”. These are properties that you would normally find in a Kafka `consumer.properties` file (without the “`consumer.`” prefix). These properties are used for by the consumer collection flows of the UMB adapter.

Please see chapter 2.5 “UMB Kafka installation” for more information on Kafka file organisation and the location of Kafka configuration files including the `producer.properties` and `consumer.properties` files.

Below is a list of all the producer properties that can be defined in the `adapter.properties` file. Any valid Kafka producer property can be defined here just by adding the “`producer.`” prefix:

- Producer basic properties:

- **producer.metadata.broker.list:** a list of Kafka broker `<host>:<port>` information used for bootstrapping knowledge about the rest of the cluster. The producer will only use it for getting metadata (topics, partitions and replicas). The socket connections for sending the actual data will be established based on the broker information returned in the metadata. Format: `host1:port1,host2:port2 ...` and the list can be a subset of brokers or a VIP pointing to a subset of brokers. Set to `localhost:9092` for example

There is no default value.

- **producer.producer.type:** This parameter specifies whether the messages are sent synchronously or asynchronously in a background thread. Valid values are:
  - `async` for asynchronous send. By setting the producer to `async` we allow batching together of requests (which is great for throughput) but open the possibility of a failure of the client machine dropping unsent data.
  - `sync` for synchronous send

Default value is `sync`.

- **producer.partitioner.class:** The partitioner class for partitioning messages amongst sub-topics. The default partitioner is based on the hash of the key:  
`kafka.producer.DefaultPartitioner`
- **producer.compression.codec:** This parameter allows you to specify the compression codec for all data generated by this producer. Valid values are `none`, `gzip` and `snappy`. The old config values work as well: 0, 1, 2 for `none`, `gzip`, `snappy` respectively.

Default value is `none`.

- **producer.compressed.topics:** This parameter allows you to set whether compression should be turned on for particular topics.
  - If the compression codec is anything other than `none`, enable compression only for specified topics if any. If the list of compressed topics is empty, then enable the specified compression codec for all topics.
  - If the compression codec is `none`, compression is disabled for all topics

Default value is `null`.

- **producer.message.send.max.retries:** This property will cause the producer to automatically retry a failed send request. This property specifies the number of retries when such failures occur. Note that setting a non-zero value here can lead to duplicates in the case of network errors that cause a message to be sent but the acknowledgement to be lost.

Default value is 3 (retries).

- **producer.retry.backoff.ms:** Before each retry, the producer refreshes the metadata of relevant topics to see if a new leader has been elected. Since leader election takes a bit of time, this property specifies the amount of time that the producer waits before refreshing the metadata.

Default value is 100 (milliseconds).

- **producer.topic.metadata.refresh.interval.ms:** The producer generally refreshes the topic metadata from brokers when there is a failure (partition missing, leader not available...). It will also poll regularly (default: every 10min so 600000ms). If you set this to a negative value, metadata will only get refreshed on failure. If you set this to zero, the metadata will get refreshed after each message sent (not recommended). Important note: the refresh happen only AFTER the message is sent, so if the producer never sends a message the metadata is never refreshed.

Default value is 600000 (milliseconds) = 10 minutes.

- Asynchronous producer properties:

- **producer.queue.buffering.max.ms:** default value 5000. Maximum time, in milliseconds, for buffering data on the producer queue when using asynchronous mode. For example a setting of 100 will try to batch together 100ms of messages to send at once. This will improve throughput but adds message delivery latency due to the buffering.
- **producer.queue.buffering.max.messages:** default value 10000. The maximum number of unsent messages that can be queued up the producer when using asynchronous mode before either the producer must be blocked or data must be dropped.
- **producer.queue.enqueue.timeout.ms:** default value -1. The amount of time to block before dropping messages when running in asynchronous mode and the buffer has reached producer.queue.buffering.max.messages.
  - If set to 0, events will be enqueued immediately or dropped if the queue is full (the producer send call will never block).
  - If set to -1 (or -X where X is a positive integer), the producer will block indefinitely and never willingly drop a send.
  - If set to +X (where X is a positive integer), the producer will block up to X milliseconds if the queue is full
- **producer.batch.num.messages:** default value 200. The number of messages to send in one batch when using asynchronous mode. The producer will wait until either this number of messages is ready to send or producer.queue.buffer.max.ms is reached.

- Synchronous producer properties:

- **producer.request.required.acks:** default value 0 (messages are acknowledged). This value controls when a produce request is considered completed. Specifically, how many other brokers must have committed the data to their log and acknowledged this to the leader? Typical values are:

- 0, which means that the producer never waits for an acknowledgement from the broker. This option provides the lowest latency but the weakest durability guarantees (some data will be lost when a server fails).
- 1, which means that the producer gets an acknowledgement after the leader replica has received the data. This option provides better durability as the client waits until the server acknowledges the request as successful (only messages that were written to the now-dead leader but not yet replicated will be lost).
- -1, which means that the producer gets an acknowledgement after all in-sync replicas have received the data. This option provides the best durability, we guarantee that no messages will be lost as long as at least one in sync replica remains.
- **producer.client.id:** default value is "". The client id is a user-specified string sent in each request to help trace calls. It should logically identify the application making the request.
- **producer.request.timeout.ms:** The amount of time the broker will wait trying to meet the producer.request.required.acks requirement before sending back an error to the client. Default value is 10000 (milliseconds) = 10 seconds.
- **producer.send.buffer.bytes:** Socket write buffer size. Default value is 100\*1024 (bytes) = 100KB.

Please see <http://kafka.apache.org/documentation.html#producerconfigs> for more details on the producer properties.

Below is a list of all the consumer properties that can be defined in the `adapter.properties` file. Any valid Kafka consumer property can be defined here just by adding the "consumer." prefix:

- **consumer.group.id:** A string that uniquely identifies the group of consumer processes to which this consumer belongs. By setting the same group id multiple processes indicate that they are all part of the same consumer group (used for balancing the consumption of messages among consumers). No default value.
- **consumer.zookeeper.connect:** a comma separated list of ZooKeeper <host>:<port> information, set to localhost:2181 for example. No default value.
- **consumer.consumer.id:** Generated automatically if not set.
- **consumer.socket.timeout.ms:** The socket timeout for network requests. The actual timeout set will be consumer.fetch.wait.max.ms + consumer.socket.timeout.ms. Default value is 30000 (milliseconds) = 30 seconds.
- **consumer.socket.receive.buffer.bytes:** The socket receive buffer for network requests. Default value is 64\*1024 (bytes) = 64 KB.
- **consumer.fetch.message.max.bytes:** The number of bytes of messages to attempt to fetch for each topic-partition in each fetch request. These bytes will be read into memory for each partition, so this helps control the memory used by the consumer. The fetch request size must be at least as

large as the maximum message size the server allows or else it is possible for the producer to send messages larger than the consumer can fetch. Default value is  $1024 * 1024$  (bytes) = 1 MB.

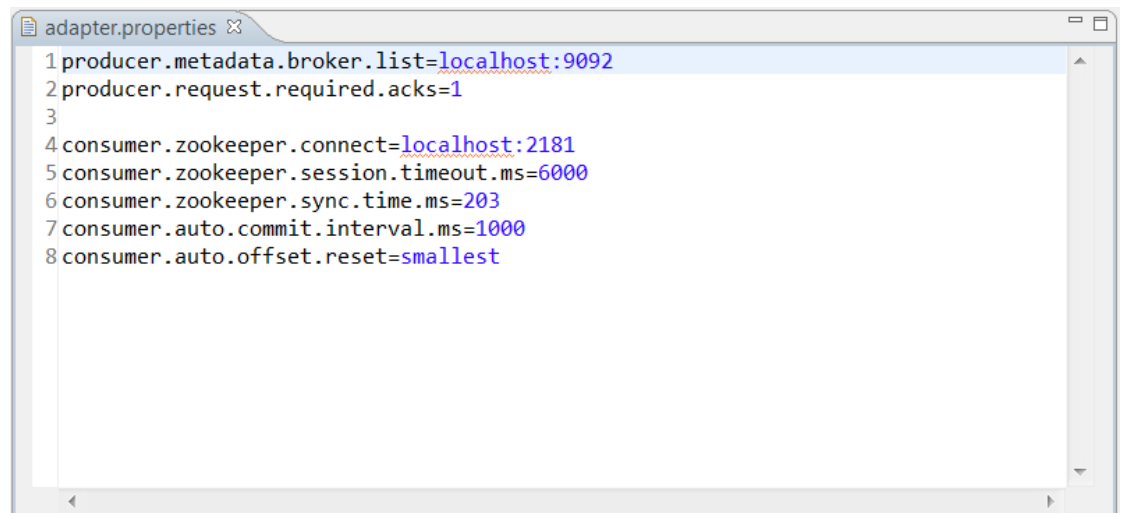
- **consumer.auto.commit.enable:** If true, periodically commit to ZooKeeper the offset of messages already fetched by the consumer. This committed offset will be used when the process fails as the position from which the new consumer will begin. Default value is `true`.
- **consumer.auto.commit.interval.ms:** The frequency in milliseconds at which the consumer offsets are committed to ZooKeeper. Default value is `60000` (milliseconds) = 60 seconds.
- **consumer.queued.max.message.chunks:** Max number of message chunks buffered for consumption. Each chunk can be up to `consumer.fetch.message.max.bytes`. Default value is `10` (chunks).
- **consumer.rebalance.max.retries:** When a new consumer joins a consumer group the set of consumers attempt to "rebalance" the load to assign partitions to each consumer. If the set of consumers changes while this assignment is taking place the rebalance will fail and retry. This setting controls the maximum number of attempts before giving up. Default value is `4` (retries).
- **consumer.fetch.min.bytes:** The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request. Default value is `1` (byte).
- **consumer.fetch.wait.max.ms:** The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy `consumer.fetch.min.bytes`. Default value is `100` (milliseconds).
- **consumer.rebalance.backoff.ms:** Backoff time between retries during rebalance. Default value is `2000` (milliseconds) = 2 seconds.
- **consumer.refresh.leader.backoff.ms:** Backoff time to wait before trying to determine the leader of a partition that has just lost its leader. Default value is `200` (milliseconds).
- **consumer.auto.offset.reset:** What to do when there is no initial offset in ZooKeeper or if an offset is out of range:
  - `smallest`: automatically reset the offset to the smallest offset
  - `largest`: automatically reset the offset to the largest offset
  - `anything else`: throw exception to the consumerDefault value is `largest`.
- **consumer.consumer.timeout.ms:** Throw a timeout exception to the consumer if no message is available for consumption after the specified interval. Default value is `-1` (never throw a timeout exception).
- **consumer.client.id:** The client id is a user-specified string sent in each request to help trace calls. It should logically identify the application making the request. Default value is the value of the `consumer.group.id` property.
- **consumer.zookeeper.session.timeout.ms:** ZooKeeper session timeout. If the consumer fails to heartbeat to ZooKeeper for this period of time it is considered dead and a rebalance will occur. Default value is `6000` (milliseconds) = 6 seconds.



- **consumer.zookeeper.connection.timeout.ms:** The max time that the client waits while establishing a connection to ZooKeeper. Default value is 6000 (milliseconds) = 6 seconds.
- **consumer.zookeeper.sync.time.ms:** How far a ZooKeeper follower can be behind a ZooKeeper leader. Default value is 2000 (milliseconds) = 2 seconds.

Please see <http://kafka.apache.org/documentation.html#consumerconfigs> for more details on the consumer properties.

Below is an example of an `adapter.properties` file:



```

1 producer.metadata.broker.list=localhost:9092
2 producer.request.required.acks=1
3
4 consumer.zookeeper.connect=localhost:2181
5 consumer.zookeeper.session.timeout.ms=6000
6 consumer.zookeeper.sync.time.ms=203
7 consumer.auto.commit.interval.ms=1000
8 consumer.auto.offset.reset=smallest
  
```

**Figure 3 - Example of adapter.properties file**

### 3.5.3 hazelcast.xml file

The `hazelcast.xml` file defines how an UMB adapter interacts with the UMB Framework with regards to Hazelcast.

This file is a standard Hazelcast XML configuration file. Please refer to Hazelcast documentation at URL: <http://hazelcast.org/documentation/> for more information on how to configure this file.

There are several sections (XML elements) that can be present in the `hazelcast.xml` file inside the root `<hazelcast>...</hazelcast>` section. Among these sections are these following (non-exhaustive list):

- A `<network>...</network>` section that defines how to connect to Hazelcast, what protocol or port to use, whether to use encryption, etc...
- An `<executor-service>...</executor-service>` section that defines the properties of the Hazelcast Executor Service used to execute actions in the UMB Framework

Other sections than the `<network>...</network>` and `<executor-service>...</executor-service>` sections are available. For a complete list of all available sections, please refer to: <http://hazelcast.org/documentation/>.

The `<network>...</network>` section defines how to connect to Hazelcast, using either<sup>5</sup>:

- IP multicast
- TCP-IP

To connect to Hazelcast using IP multicast, you need to define a `<multicast>...</multicast>` section inside the `<join>...</join>` section of the `<network>...</network>` section. For example:

```
<multicast enabled="true">
    <multicast-group>224.2.2.3</multicast-group>
    <multicast-port>54327</multicast-port>
</multicast>
```

To connect to Hazelcast using TCP-IP, you need to define a `<tcp-ip>...</tcp-ip>` section inside the `<join>...</join>` section of the `<network>...</network>` section. For example:

```
<tcp-ip enabled="true">
    <interface>localhost</interface>
</tcp-ip>
```

The `<network>...</network>` section also defines what ports to use in the `<port>...</port>` section. By default, Hazelcast cluster members (each UMB Adapter is a Hazelcast cluster member) use port numbers starting at 5701: 5701, 5702, 5703, ...:

```
<port auto-increment="true" port-
count="100">5701</port>
```

The `<executor-service>...</executor-service>` section defines the properties of the Hazelcast Executor Service used to execute actions in the UMB Framework. In this section you can define how many threads are used for processing actions by the UMB Adapter by setting the value of the `<pool-size>...</pool-size>` XML element. You can also define the size of the action request queue by setting the value of the `<queue-capacity>...</queue-capacity>` XML element:

```
<executor-service name="default">
    <pool-size>7</pool-size>
    <!--Queue capacity. 0 means Integer.MAX_VALUE.-->
    <queue-capacity>0</queue-capacity>
</executor-service>
```

---

<sup>5</sup> It is also possible to connect using Amazon Web Services (for connecting to Amazon Cloud Services)

Below is an example of a `hazelcast.xml` file:

```
1 <?xml version="1.0" encoding="UTF-8"?>
3* ~ Copyright (c) 2008-2013, Hazelcast, Inc. All Rights Reserved.
17
18 <hazelcast xsi:schemaLocation="http://www.hazelcast.com/schema/config hazelcast-config-3.2.xsd"
19           xmlns="http://www.hazelcast.com/schema/config"
20           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
21   <group>
22     <name>dev</name>
23     <password>dev</password>
24   </group>
25   <management-center enabled="false">http://localhost:9080/mancenter</management-center>
26   <network>
27     <port auto-increment="true" port-count="100">5701</port>
28     <outbound-ports>
29       <!--
30         Allowed port range when connecting to other nodes.
31         0 or * means use system provided port.
32       -->
33     <ports>0</ports>
34   </outbound-ports>
35   <join>
36     <multicast enabled="false">
37       <multicast-group>224.2.2.3</multicast-group>
38       <multicast-port>54327</multicast-port>
39     </multicast>
40     <tcp-ip enabled="true">
41       <interface>localhost</interface>
42     </tcp-ip>
43     <aws enabled="false">
44       <access-key>my-access-key</access-key>
45       <secret-key>my-secret-key</secret-key>
46       <!--optional, default is us-east-1 -->
47       <region>us-west-1</region>
48       <!--optional, default is ec2.amazonaws.com. If set, region shouldn't be set as i
```

Figure 4 - Example of `hazelcast.xml` file

### 3.5.4 log4j.xml file

The `log4j.xml` file is the Log4J configuration file for the whole UMB adapter application. It is a standard Apache Log4J configuration file.<sup>6</sup>

This file contains three main sections where the following items are defined:

- **Appenders:** appenders mainly define where the log messages are sent, and the pattern used for logging the messages. There is one appenders defined by default:
  - **CONSOLE:** for logging to the console
  - **FILE:** for logging to the `${UMB_adapter name in upper case}ADAPTER_DATA}/logs/<adapter name>-adapter.log` file (for example: `/var/opt/UMB/file-adapter/logs/file-adapter.log` for the File adapter)

If you want to log to a file for example, you can add an appender to do just this.

- **Loggers:** loggers are defined by Java package names. Each logger defines its own log level and appender references.
- **Root:** the root section defines the default log level and the default appender references to use for logging

<sup>6</sup> Please see <http://logging.apache.org/log4j/1.2/> to learn more about Apache Log4J configuration files.

You can make your own changes to the `log4j.xml` file, for example:

- Modifying existing appenders or creating new ones
- Modifying existing loggers: changing the log level or the appender references
- Adding new loggers, for 3<sup>rd</sup> party products for example
- Modifying the default log level and appender references in the root section of the file

Once you have made changes to the `log4j.xml` file, you will need to restart the either need to restart UMB adapter.

Log files are stored in the `${UMB_adapter name in upper case}>ADAPTER_DATA}/logs` directory (for example: `/var/opt/UMB/file-adapter/logs` for the File adapter).

## 3.6 TeMIP Adapter Specific configuration

The TeMIP Adapter provides action and collection flow services to/from TeMIP.

As a Flow producer, the adapter will:

- Respond to collection flow actions for dynamic flows (`CreateFlow`, `DeleteFlow`, `ResynchFlow`) and static flows (`ResynchFlow`) from other UMB Adapters
- Automatically start static flows defined in the TeMIP Adapter's configuration file: `AdapterConfiguration.xml`

As an action services provider, the adapter will:

- Respond to action execution requests from other UMB Adapters, i.e.:
  - Alarm Object directives
  - Trouble Ticket directives
  - Any directive using the `PassthroughAction`

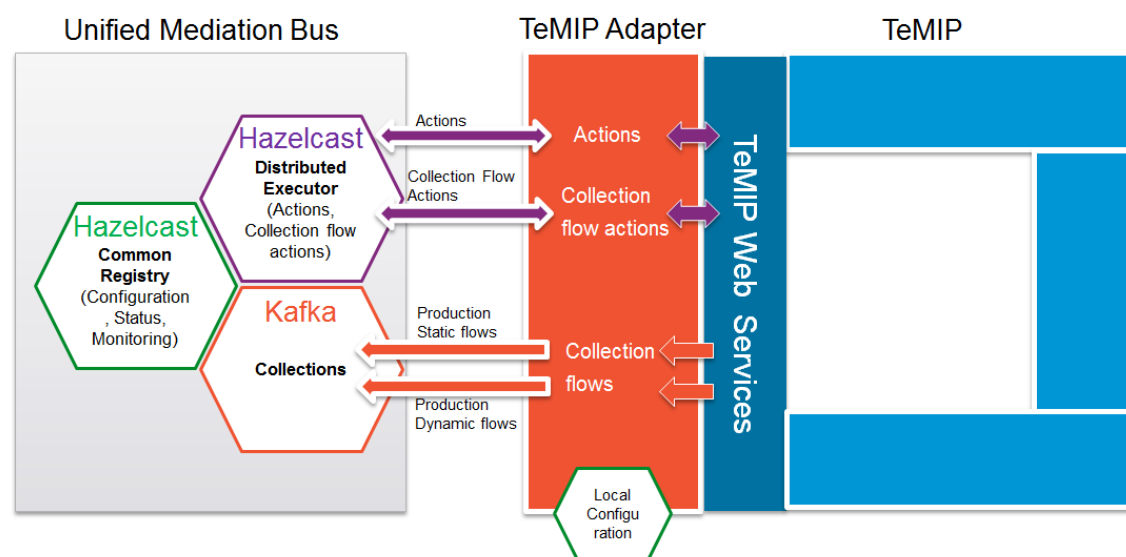
The TeMIP Adapter is composed of:

- An Adapter start script
- Configuration files:
  - The Adapter properties file: `adapter.properties` that defines properties for the adapter including connection information for Kafka/ZooKeeper
  - The Adapter's Hazelcast configuration file: `hazelcast.xml` that defines how to connect to the UMB Hazelcast Central Repository
  - The Adapter's Log4j configuration file: `log4j.xml`
  - The Adapter configuration file: `AdapterConfiguration.xml` that defines the flows and actions provided by the adapter
  - One or more TeMIP configuration files: named `TeMIP_configuration.xml` or not that defines the connection to

TeMIP. TeMIP configuration files are associated with collection flows in the `AdapterConfiguration.xml` file. Each collection flow can use a different TeMIP configuration file or all flow can use the same file or anything in between.

- One or more `axis2.xml` (the name and path of the file is configurable in the `TeMIP_configuration.xml` file) TWS configuration files that define the configuration for connecting to TWS. Each `TeMIP_configuration.xml` file can have a different TWS configuration file.
- Library files that define the Adapter's behavior

The following figure explains the overall architecture of the TeMIP Adapter.



**Figure 5 - TeMIP adapter overview**

In the above figure, the TeMIP Adapter is used to produce alarm collection flows from TeMIP to the Unified Mediation Bus. Both static and dynamic flows are supported. The TeMIP Adapter can also respond to collection flow actions and TeMIP Alarm Object directives actions from UMB.

The following sections will explain how to configure the TeMIP Adapter.

### 3.6.1 TeMIP Adapter Installation

Please refer to the [R1] *Unified Mediation Bus installation and configuration Guide* for details on how to install the TeMIP Adapter: see Chapter 3 “Unified Mediation Bus Adapters”.

### 3.6.2 TeMIP Adapter Configuration

The configuration files of the TeMIP Adapter are located in the `<TeMIP Adapter installation directory>/conf` folder. Each of the configuration files is explained in detail below.

### 3.6.2.1 The adapter.properties file

The Adapter properties file: `adapter.properties` defines properties for the adapter including connection information for the UMB Kafka/ZooKeeper instance(s).

The following properties are defined by default in this file:

- **producer.metadata.broker.list:** a list of Kafka broker `<host>:<port>` information. Set to `localhost:9092` by default
- **producer.request.required.acks:** set to `1` by default, indicating that Kafka is in a mode where messages are acknowledged
- **consumer.zookeeper.connect:** a list of ZooKeeper `<host>:<port>` information. Set to `localhost:2181` by default
- **consumer.zookeeper.session.timeout.ms:** set to `6000` by default
- **consumer.zookeeper.sync.time.ms:** set to `203` by default
- **consumer.auto.commit.interval.ms:** set to `1000` by default
- **consumer.auto.offset.reset:** set to `smallest` by default
- **uca.collection.rawCollectionQueueSize:** size of the raw event collection queue used to store raw events during resynchronization. Set to `10000` by default

Please refer to the [R1] *Unified Mediation Bus installation and configuration Guide* for details on how to configure the `adapter.properties` file.

### 3.6.2.2 The hazelcast.xml file

The Adapter's Hazelcast configuration file: `hazelcast.xml` defines how to connect to the UMB Hazelcast instance(s).

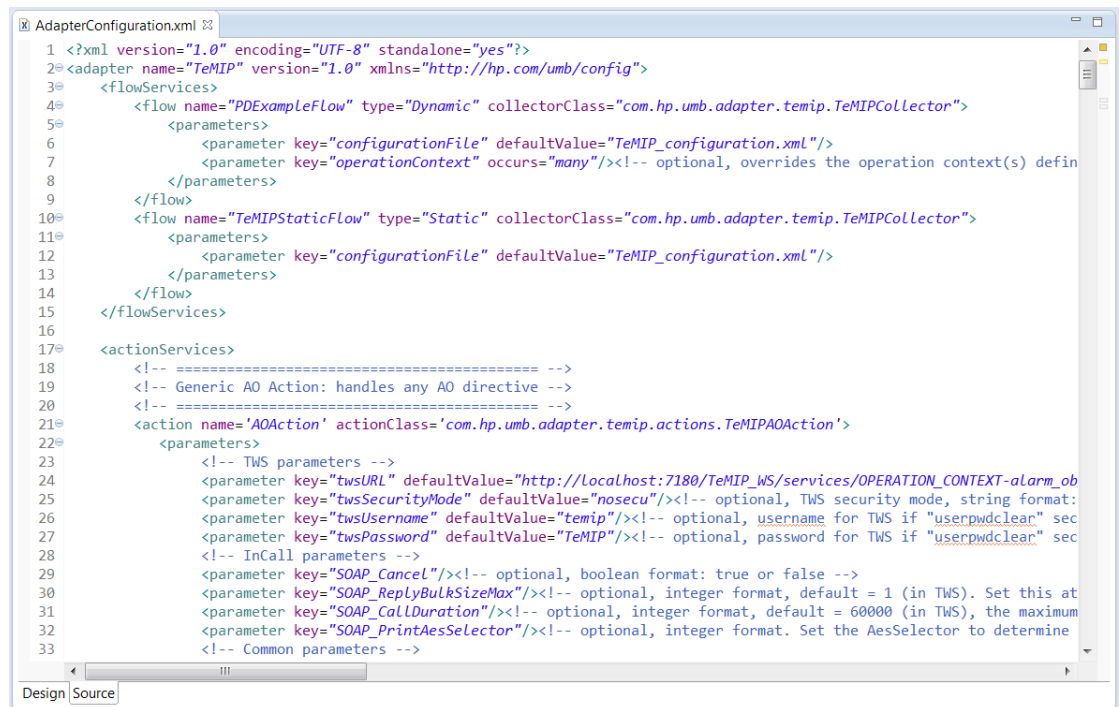
Please refer to the [R1] *Unified Mediation Bus installation and configuration Guide* for details on how to configure the `hazelcast.xml` file.

### 3.6.2.3 The log4j.xml file

The Adapter's Log4j configuration file: `log4j.xml`

### 3.6.2.4 The AdapterConfiguration.xml file

The Adapter configuration file: `AdapterConfiguration.xml` defines the flows and actions provided by the adapter.



**Figure 6 - The TeMIP Adapter's AdapterConfiguration.xml file**

In the `<flowServices>...</flowServices>` section, the collection flows produced by the TeMIP Adapter must be declared. Collection flows can either be static or dynamic.

By default, one static flow is defined that uses the `TeMIP_configuration.xml` file. This file contains connection information to TeMIP as well as the list of Operation Contexts that are part of the collection flow, some options, and the list of Custom Alarm Object fields to use.

Flow collections (both dynamic and static flows) can be added or removed by editing the `<flowServices>...</flowServices>` section.

Please refer to the [R1] *Unified Mediation Bus - Adapter Development Guide* for learning how to add producer collection flow services.

In the `<actionServices>...</actionServices>` section, the action services proposed by the TeMIP Adapter are declared. These services include action services for executing:

- Alarm Object directives using the AOAction action (that can process any AO directive except GETCHILDRENALARMS, GETPARENTALARMS, GETVARSELECTOR, or SUMMARIZE which are not yet implemented) or any of the Directive specific actions:
  - AOAction\_Acknowledge
  - AOAction\_Addparent
  - AOAction\_Archive
  - AOAction\_Clearalarm
  - AOAction\_Close
  - AOAction\_Create
  - AOAction\_Delete

- AOAction\_Demote
- AOAction\_Dump
- AOAction\_Getevent
- AOAction\_Groupalarms
- AOAction\_Handle
- AOAction\_Release
- AOAction\_Removeparent
- AOAction\_Renameentity
- AOAction\_Set
- AOAction\_Show
- AOAction\_Terminate
- AOAction\_Unacknowledge
- AOAction\_Unclearalarm
- AOAction\_Undotermiante
- AOAction\_Ungroupalarms
- AOAction\_Ungroupall
- Trouble Ticket directives using the TTACTION action (that can process any TT directive) or any of the Directive specific actions:
  - TTACTION\_AssociateTT
  - TTACTION\_CancelTT
  - TTACTION\_ClearAll
  - TTACTION\_CloseTT
  - TTACTION\_Create
  - TTACTION\_CreateTT
  - TTACTION\_Delete
  - TTACTION\_Deregister
  - TTACTION\_Directory
  - TTACTION\_DisplayAssociatedTt
  - TTACTION\_DisplayRelatedTt
  - TTACTION\_DissociateTT
  - TTACTION\_RebuildAll
  - TTACTION\_Register
  - TTACTION\_Set
  - TTACTION\_Show
  - TTACTION\_Synchronize
  - TTACTION\_Test
- Any directive using the PassthroughAction action

In order to execute an AOAction for example, the following code can be used:



```
// Execute AO action on TeMIP Adapter
List<Parameter> parameters = new ArrayList<Parameter>();
parameters.add(new Parameter("directiveName", "SHOW"));
parameters.add(new Parameter("entityName",
    "operation_context my_oc alarm_object 15452"));
parameters.add(new Parameter("UserId", "user1"));
ActionQuery actionQuery = new ActionQuery(localAdapter, "AOAction",
    parameters);
actionQuery.setActionId("00000001");
ActionReply actionReply = null;

try {
    actionReply = actionQuery.executeSyncAction("TeMIP");
} catch (IllegalActionStateException e) {
    Log.error(String
        .format("Unexpected exception thrown executing action (id = %s): %s",
            actionQuery.getActionId(),
            e.getLocalizedMessage()));
}
```

**Figure 7 - Executing an Alarm Object directive action on TeMIP Adapter**

The above screen capture shows how to execute an Alarm Object directive on TeMIP Adapter when TWS is in the “No Security” mode (the default mode).

To execute a TTACTION, the following code can be used:

```
// Execute TT action on TeMIP Adapter in TWS UserPwdClear mode
List<Parameter> parameters = new ArrayList<Parameter>();
parameters.add(new Parameter("directiveName", "CREATE"));
parameters.add(new Parameter("entitySpec", "TT_SERVER noc_ns:.SM"));
parameters.add(new Parameter("Application_Context", "AC"));
parameters.add(new Parameter("Application_Dn", "AD"));

// TWS Parameters
parameters.add(new Parameter("twSecurityMode", "userpwdclear"));
parameters.add(new Parameter("twUsername", "temip"));
parameters.add(new Parameter("twPassword", "TeMIP"));

ActionQuery actionQuery = new ActionQuery(localAdapter,
    "TTAction", parameters);
actionQuery.setActionId("00000001");
ActionReply actionReply = null;

try {
    actionReply = actionQuery.executeSyncAction("TeMIP");
} catch (IllegalActionStateException e) {
    fail(String
        .format("Unexpected exception thrown executing action (id = %s): %s",
            actionQuery.getActionId(),
            e.getLocalizedMessage()));
}
```

**Figure 8 - Executing a Trouble Ticket directive action on TeMIP Adapter**

The above screen capture shows how to execute a Trouble Ticket directive on TeMIP Adapter when TWS is in the “User Password Clear” mode.

To execute a PassthroughAction, the following code can be used:

```
// Execute Pass-through action on TeMIP Adapter
List<Parameter> parameters = new ArrayList<Parameter>();

ActionQuery actionQuery = new ActionQuery(localAdapter,
    "PassthroughAction", parameters);
actionQuery.setActionId("00000001");
actionQuery.setRawData("<oper:Create_Request xmlns:oper=\"http://operation_context-alarm_object.types.ws.temip.ov.hp.com\">"
    + "<oper:EntitySpec><oper:Natural>OPERATION_CONTEXT .wandl_1_test_oc_alarm_object 1</oper:Natural></oper:EntitySpec>"
    + "<oper:Arguments><oper:Managed_Object><oper:Natural>WANDL wandl_1</oper:Natural></oper:Managed_Object>"
    + "<oper:Alarm_Type>EquipmentAlarm</oper:Alarm_Type><oper:Event_Time><oper:Year>2012</oper:Year><oper:Month>12</oper:Month>"
    + "<oper:Day>21</oper:Day><oper:Hour>21</oper:Hour><oper:Minute>21</oper:Minute><oper:Second>21</oper:Second>"
    + "<oper:Millisecond>21</oper:Millisecond></oper:Event_Time><oper:Probable_Cause>Unknown</oper:Probable_Cause>"
    + "<oper:Perceived_Severity>Major</oper:Perceived_Severity><oper:Backed_Up_Status>true</oper:Backed_Up_Status>"
    + "<oper:Notification_Identifier>1</oper:Notification_Identifier>"
    + "<oper:Additional_Text>Communication Failure on SubSystem B</oper:Additional_Text></oper:Arguments></oper:Create_Request>");
ActionReply actionReply = null;

try {
    actionReply = actionQuery.executeSyncAction("TeMIP");
} catch (IllegalActionStateException e) {
    fail(String
        .format("Unexpected exception thrown executing action (id = %s): %s",
            actionQuery.getActionId(),
            e.getLocalizedMessage()));
}
```

**Figure 9 - Executing a Passthrough action on TeMIP Adapter**

The above screen capture shows how to execute a Passthrough action on TeMIP Adapter. With Passthrough actions, the XML code of the directive to execute is put directly in the rawData of the actionQuery.

Please refer to the [R1] *Unified Mediation Bus installation and configuration Guide* for details on how to configure the AdapterConfiguration.xml file.

### 3.6.2.5 The TeMIP configuration file(s)

The TeMIP configuration file: usually named TeMIP\_configuration.xml, if there is only one such file, defines how to connect to TeMIP. Each collection flow defines which TeMIP configuration file to use: this is done in the AdapterConfiguration.xml file. As a consequence, you can possibly have multiple TeMIP configuration files in the TeMIP Adapter.



**Figure 10 - Example of a TeMIP configuration file**

Each TeMIP configuration file contains the same type of information. It is necessary to configure the TeMIP configuration file to fit both your needs and your TeMIP configuration. The main sections are the following:

### Authentication information

In case TWS is setup in either “User Password Clear” or “User Password Encrypted” security modes (Please see TWS documentation for more information on how to set up TWS in either of these modes), the

`<Authentication>...</Authentication>` section allows you to configure the username and password to use for connecting to TWS:

- **UserName:** by default this element is set to *temip*. Please set this username to a valid unix account username to be used for connecting to TWS. The unix account has to be valid on the server hosting TWS.
- **Password:** by default this element is set to *TeMIP*. Please set this password to the correct password of the unix account mentioned in the `<UserName>...</UserName>` section.

If TWS is setup in the “No Security” security mode, then the username and password listed in this section are not used for connecting to TWS, since the “No Security” mode does not use usernames and passwords.

### Axis information

The `<Axis>...</Axis>` section contains the path to the `axis2.xml` file associated with the `TeMIP_configuration.xml` file being configured:

- **XmlPath:** by default this element is set to *conf/axis2.xml*.

### TeMIP/TWS information

In order to properly configure its connection to TeMIP, a TeMIP collection flow must be told which TeMIP director to connect to. This is done by configuring

the `<DirectorConfiguration>...</DirectorConfiguration>` section of the TeMIP configuration file. Please validate that the information inside this section is correct with regards to your TeMIP/TWS setup.

You should pay special attention to the following elements:

- **MachineName:** by default this element is set to *localhost*. If your TeMIP director is not located on the local host, please update the value of this element accordingly with the host name (or IP address) of your TeMIP director
- **TeMIPDirectorEntity:** Please verify that the value of this element matches your TeMIP director entity name (execute `manage show temip "*"` on your TeMIP director host to get the TeMIP director entity name)
- **TWSServerPort:** this element is set to *7180* by default, which is the default TWS port number. Please verify that the value of this element matches your TWS configuration

Below is an example of the  
`<DirectorConfiguration>...</DirectorConfiguration>` section of the  
*TeMIP\_configuration.dynamic.xml* file:

```
<DirectorConfiguration>
  <MachineName>mytemip.mycompany.com</MachineName>
  <!-- Put here TeMIP director name.
        If you leave this field as is, dynamic flows
        operations will not work -->
  <TeMIPDirectorEntity>.temip.mytemip_temip</TeMIPDirectorEntity>
  <TWSServerPort>7180</TWSServerPort>
</DirectorConfiguration>
```

**Figure 11 - Example of TeMIP/TWS configuration in the  
TeMIP\_configuration.dynamic.xml file**

Below is an example of how to verify the TeMIP director entity name of your TeMIP  
director (the command below has to be executed on the TeMIP director host):

```
# manage show temip "*"
TeMIP Framework (V6.0.0)

Using default ALL IDENTIFIERS

TEMIP mytemip_ns:.temip.mytemip_temip
On director: mytemip_ns:.temip.mytemip_director
AT Wed, Aug 7, 2013 03:17:31 PM Identifiers

Examination of attributes shows
                                TeMIP Name =
mytemip_ns:.temip.mytemip_temip
```

**Figure 12 - How to query the TeMIP director entity name**

## Operation Context information

In order to properly configure a TeMIP collection flow, the list of Operation Contexts  
to use as part of the collection flow is needed. This can be specified in the  
`<OperationContexts>...</OperationContexts>` section:

```
</temip_ws:TeMIP_WS>
<acs_ws:ACS_WS xmlns="http://acs_ws.temip_ws.temip.hp.com">
  <OperationContexts>
    <OperationContext>my_oc</OperationContext>
  </OperationContexts>
  <AggregateEvent>true</AggregateEvent>
  <!-- Specifies SourceScope for all Sources of ACS Collections.
        Valid values are the following:
        NOT_TERMINATED - default value
        OUTSTANDING
        NOT_HANDLED
```

**Figure 13 - How to specify the Operation Context(s) in the TeMIP configuration  
file**

It is possible to specify more than just one Operation Context.

The TeMIP Adapter processes only Aggregate Events as specified by the `<AggregateEvent>true</AggregateEvent>` section. Please make sure that the Operation Contexts that you list in the `<OperationContexts>...</OperationContexts>` section are defined to emit aggregate events, otherwise the TeMIP Adapter won't work properly.

### Alarm Object custom fields

The TeMIP configuration file is also the place where you need to declare any Alarm Object custom attributes that you want the TeMIP Adapter to use. Alarm collections created by the TeMIP Adapter will contain all the custom fields declared in the TeMIP configuration file.

Alarm Object custom attributes are to be declared in the `<CustomAttributes>...</CustomAttributes>` section of the file.

To add a new Alarm Object custom attributes, you need to add a `<CustomAttribute>...</CustomAttribute>` section inside the `<CustomAttributes>...</CustomAttributes>` section of the file.

```

</CustomAttribute>
<CustomAttribute>
  <Attribute>Correl Notif Info</Attribute>
  <Datatype>XmlString</Datatype>
</CustomAttribute>
<CustomAttribute>
  <Attribute>My Custom Attribute</Attribute>
  <Datatype>XmlString</Datatype>
</CustomAttribute>
</CustomAttributes>
<QueueSize>1000</QueueSize>
<!--PassingClasses>
  <ClassHierarchy>
    <Class>NSI_SYSTEM</Class>

```

**Figure 14 - Adding a custom AO attribute in the TeMIP\_configuration.dynamic.xml file**

### 3.6.2.6 The axis2.xml configuration file(s)

The AXIS2 configuration file: usually named `axis2.xml`, if there is only one such file, defines how to connect to TWS.

Each TeMIP configuration file defines which `axis2.xml` file to use: this is done in the `TeMIP_configuration.xml` file. As a consequence, you can possibly have multiple AXIS2 configuration files in the TeMIP Adapter.



**Figure 15 - Example of an axis2.xml configuration file**

Each AXIS configuration file contains the same type of information. It is usually not necessary to modify this file, unless you want to modify the security mode used to connect to TWS. By default, the `axis2.xml` file delivered with the TeMIP Adapter is set to the “No Security” mode. The list of possible security modes is the following:

- “No Security” mode
- “User Password Clear” mode
- “User Password Encrypted” mode

There are several sections in this file. The section that deals with the TWS security mode is the `OutflowSecurity` parameter section. In “No Security” mode, the `<items>...</items>` section is empty: `<items></items>`, whereas in “User Password Clear” mode, the `<items>...</items>` section contains “UsernameToken”: `<items>UsernameToken</items>`

The “User Password Encrypted” mode is not yet supported by the TeMIP Adapter.

## 3.7 OSSAF Adapter

The OSSAF Adapter provides actions services to the OSS Analytics Foundation (OSSAF). It translates UMB actions into calls to the OSSAF REST API. The results of such calls are given back to the caller through a specific class `com.hp.umb.ossaf.api.Reply`, which is provided in a separate jar file.

### 3.7.1 Specific properties

The OSSAF Adapter configuration file `adapter.properties` defines all properties supported by the OSSAF Adapter.

In particular, properties below are specific:

<code>url.connection.timeout</code>	Defines the timeout value, in milliseconds, to use when establishing the connection with the OSSAF server. Default is 10000.
<code>url.connection.retry</code>	Defines the maximum number of connections to attempt for establishing the connection with the OSSAF server.

### 3.7.2 Specific configuration

The OSSAF Adapter configuration file `AdapterConfiguration.xml` defines all actions supported by the OSSAF Adapter.

It is up to the project integrator to correctly set up this configuration file to fit the project specific needs.

#### 3.7.2.1 Configuring a query fact values request

The OSSAF adapter defines a generic query for such purpose:

```
<action name="queryFactValues" inherits="query"
        actionClass="com.hp.umb.ossaf.adapter.Actions">
```

This query should not be changed. Integrator should define more specific queries for OSSAF Adapter client needs by inheriting this query and setting particular parameters for its DB.

For example:

```
<action name="queryTest1" inherits="queryFactValues"
        actionClass="com.hp.umb.ossaf.adapter.Actions">
  <parameters>
    <parameter key="fact" mandatory="true" occurs="many"
      defaultValue="ACKDURATION" />
    <parameter key="dim" occurs="many"
      defaultValue="IDENTIFIER/SEVERITYNAME" />
    <parameter key="begin" mandatory="true"
      defaultValue="20150402-120000"/>
    <parameter key="end" defaultValue="20150404-000000"/>
    <parameter key="offset" defaultValue="0"/>
    <parameter key="batchsize" defaultValue="100"/>
    <parameter key="aggreg" defaultValue="0"/>
  </parameters>
</action>
```

### 3.7.2.2 Configuring a query distinct dimension values request

The OSSAF adapter defines a generic query for such purpose:

```
<action name="queryDistinctDimensionValues"
        inherits="queryDimensionValues"
        actionClass="com.hp.umb.ossaf.adapter.Actions">
```

This query should not be changed. Integrator may define more specific queries for OSSAF Adapter client needs by inheriting this query and setting particular parameters for its DB.

For example:

```
<action name="queryTest3"
        inherits="queryDistinctDimensionValues"
        actionClass="com.hp.umb.ossaf.adapter.Actions">
  <parameters>
    <parameter key="dim" defaultValue="PROBABLECAUSENAME" />
    <parameter key="offset" defaultValue="0"/>
    <parameter key="aggreg" defaultValue="0"/>
  </parameters>
</action>
```

### 3.7.2.3 Configuring a query matching dimension values request

The OSSAF adapter defines a generic query for such purpose:

```
<action name="queryMatchingDimensionValues"
```

```
inherits="queryDimensionValues"
actionClass="com.hp.umb.ossaf.adapter.Actions">
```

This query should not be changed. Integrator may define more specific queries for OSSAF Adapter client needs by inheriting this query and setting particular parameters for its DB.

For example:

```
<action name="queryTest4"
inherits="queryMatchingDimensionValues"
actionClass="com.hp.umb.ossaf.adapter.Actions">
<parameters>
<parameter key="dim" defaultValue="DOMAINNAME" />
<parameter key="pattern" defaultValue="*ossv040*" />
<parameter key="batchsize" defaultValue="100"/>
<parameter key="aggreg" defaultValue="0"/>
</parameters>
</action>
```

#### 3.7.2.4 Configuring a query for dimension value lookup request

The OSSAF adapter defines a generic query for such purpose:

```
<action name="queryLookupDimensionValue"
inherits="query"
actionClass="com.hp.umb.ossaf.adapter.Actions">
```

This query should not be changed. Integrator may define more specific queries for OSSAF Adapter client needs by inheriting this query and setting particular parameters for its DB.

For example:

```
<action name="queryTest5"
inherits="queryLookupDimensionValue"
actionClass="com.hp.umb.ossaf.adapter.Actions">
<parameters>
<parameter key="dim" defaultValue="IDENTIFIER" />
<parameter key="keyValue" defaultValue="hello" />
<parameter key="aggreg" defaultValue="0"/>
</parameters>
</action>
```

#### 3.7.2.5 Configuring OSSAF Adapter on multiple servers

OSSAF Adapter has the capability to be deployed on multiple servers, as it is by default configured to use the grouping mechanism of UMB adapters.

All OSSAF adapters (within a same hazelcast cluster) are indeed reachable by targeting the "OSSAF" name.

But as any UMB adapter should require a unique name, you will have to change the name of your OSSAF adapters manually. This is done by modifying the attribute name of the adapter (by default it is "OSSAF-x").



```
<adapter name="OSSAF-x" actionGroup="OSSAF" ...
```

### 3.7.3 Deploying OSSAF Adapter as an EJB

OSSAF Adapter comes with an EJB packaged in  
\$UMB\_OSSAFADAPTER\_HOME/lib/umb-ossaf-ejb-<version>.jar.

The above jar file is an EJB 3.1 compliant and requires a Java EE 6 platform to be deployed on.

In the specific case of JBoss AS 7.x or JBoss EAP 6.x, OSSAF Adapter package brings the JBoss Modules needed by the OSSAF Adapter EJB, available under  
\$UMB\_OSSAFADAPTER\_HOME/jboss/modules.

To deploy the OSSAF Adapter EJB on JBoss EAP 6.x:

- Copy all modules into your \$JBASS\_HOME/modules, for instance:

```
# cp -r $UMB_OSSAFADAPTER_HOME/jboss/modules/*  
$JBASS_HOME/modules
```

- Copy EJB into your JBoss server deployment directory, for instance:

```
# cp $UMB_OSSAFADAPTER_HOME/lib/umb-ossaf-ejb-  
<version>.jar $JBASS_HOME/server/deployments
```

That's it. The OSSAF Adapter EJB has an automatic startup capability when it's deployed on Java EE 6.

### 3.7.4 Writing an OSSAF Adapter client

In order to develop an OSSAF Adapter client, user needs to have the OSSAF API in its classpath in order to understand the object returned by the action call.

Such object is returned through the rawData field of the UMB ActionReply.

It is defined in umb-ossaf-api.jar part of the libraries delivered with OSSAF Adapter.

Example of code of a Junit test:

```
import com.hp.umb.ossaf.api.Reply;  
  
public class MyClientClassTest {  
  
    ...  
  
    @Test  
    public void testQueryTest1AndReply() throws Exception {  
  
        ActionQuery actionQuery = new ActionQuery(adapter, "OSSAF",  
            "queryTest1");  
        actionQuery.setActionId("0001");  
  
        ActionReply actionReply = null;  
        try {  
            actionReply = actionQuery.executeSyncAction();  
        } catch (Exception e) {
```

```

        fail(String.format("Unexpected exception thrown executing action
(id = %s): %s", actionQuery.getActionId(), e.getLocalizedMessage()));
    }
    assertNotNull(actionReply);
    assertEquals(ActionStatus.SUCCESS, actionReply.getStatus());
    Reply reply = (Reply) actionReply.getRawData();
    assertNotNull(reply);
    Log.debug("REPLY=" + reply);
}

```

---

#### Note

- When delivering an UCA-EBC Value Pack that brings such OSSAF Adapter client code, integrator will need to put umb-ossaf-api.jar under \$UCA\_EBC\_HOME/externallib directory (if multiple Value Packs use it) or directly packaged within the Value Pack libraries.
- 

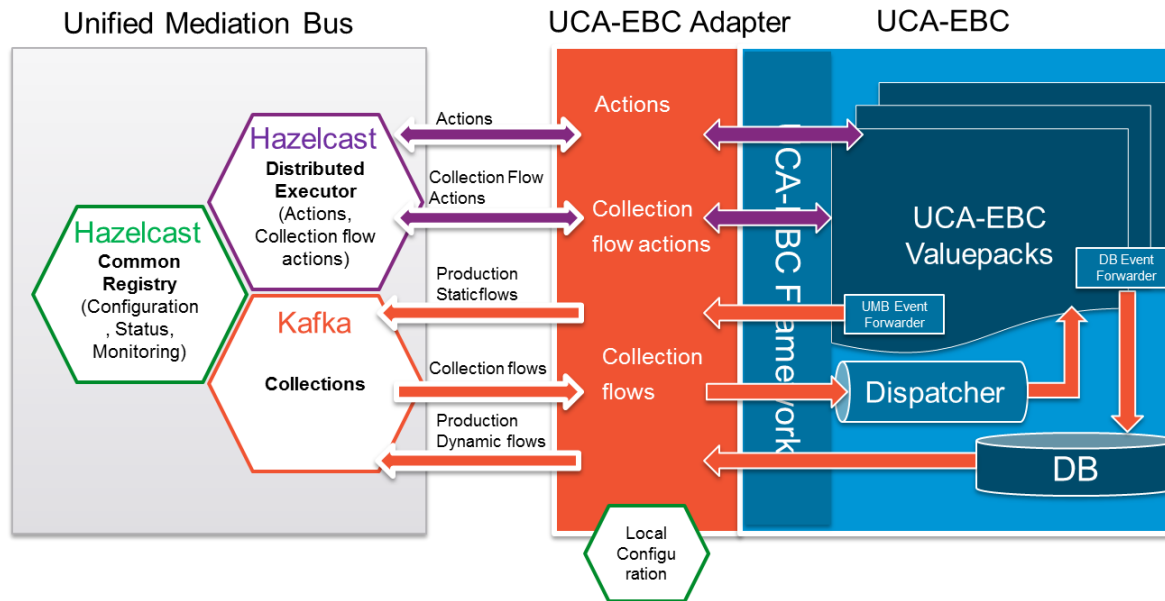
## 3.8 UCA-EBC Adapter specific configuration

The UCA-EBC Adapter is completely embedded with the UCA-EBC application starting from version V3.3. This means the Adapter runs in the same JVM as the UCA-EBC process. The adapter part is instantiated at UCA-EBC startup.

The UCA-EBC Adapter, allows UCA-EBC

- producing events to Static Unified Mediation Bus flows, thanks to a specific UMB Alarm Forwarder,
- producing events to Dynamic Unified Mediation Bus flows through the UCA Event DB facility,
- collecting events from Unified Mediation Bus flows,
- performing actions through the Unified Mediation Bus actions service.

This can be represented as follow:



**Figure 16 - UCA-EBC adapter architecture**

### 3.8.1 UCA-EBC Adapter Configuration

The configuration requirements of the UCA-EBC Adapter are the same as any other adapters. It requires a properties file to set the Adapter properties, the Hazelcast.xml file for the Common registry access, and the AdapterConfiguration.xml file to define the provided services.

All the requested configuration files are searched in the UCA-EBC configuration directory: \${UCA\_EBC\_DATA}/conf.

#### 3.8.1.1 The properties file

Has the UCA-EBC UMB Adapter is embedded in the UCA-EBC application, there no specific adapter.properties file for this adapter. Instead the properties are defined in the standard uca-ebc.properties file.

The following properties are defined by default in this file as follow:

```
#####
##
#                               UMB Mediation properties
use.new.generation.adapter=true

# UMB Consumer properties
consumer.zookeeper.connect=localhost:2181
consumer.zookeeper.session.timeout.ms=6000
consumer.zookeeper.sync.time.ms=203
consumer.auto.commit.interval.ms=1000
consumer.auto.offset.reset=smallest

# UMB Consumer properties
producer.metadata.broker.list=localhost:9092
producer.request.required.acks=1
#####
##
```

Please refer to the [R1] *Unified Mediation Bus installation and configuration Guide* for details on how to configure the Adapter's properties.

### 3.8.1.2 The hazelcast.xml file

The Adapter's Hazelcast configuration file: `hazelcast.xml` defines how to connect to the UMB Hazelcast instance(s).

Please refer to the [R1] *Unified Mediation Bus installation and configuration Guide* for details on how to configure the `hazelcast.xml` file.

### 3.8.1.3 The logging configuration file

The Adapter's Log4j configuration is done through the standard UCA-EBC configuration file: `uca-ebc-log4j.xml`

### 3.8.1.4 The AdapterConfiguration.xml file

The Adapter configuration file: `AdapterConfiguration.xml` defines the Event flows that UCA-EBC provides.

Has for any other UMB Adapters the `AdapterConfiguration.xml` file defines the adapter name (by default set to "UCA-EBC"). This name must be changed if the solution is made of several UCA-EBC servers.

### Defining static flows:

For static Flows the `collectorClass` must be set to:

`com.hp.uca.expert.mediation.adapter.UcaStaticCollector`

No flow parameters need to be defined.

Here is an example of Static Flow Service definitions for UCA-EBC:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<adapter name="UCA-EBC" version="1.0"
xmlns="http://hp.com/umb/config">
  <flowServices>
    <flow name="UcaStaticForwarderFlow" type="Static"
collectorClass="com.hp.uca.expert.mediation.adapter.UcaStaticCo
llector">
      </flow>
    <flow name="UcaStaticEventForwarderFlow" type="Static"
collectorClass="com.hp.uca.expert.mediation.adapter.UcaStaticCo
llector">
      </flow>
    </flowServices>
  </adapter>
```

---

#### Note

The static flows provided by UCA-EBC do not support resynchronization.

---

### Defining dynamic flows:

For dynamic Flows the `collectorClass` must be set to the right collector class extending

`com.hp.uca.expert.mediation.adapter.UcaDynamicCollector`

The flow parameters need to be defined.

Here is an example of a Dynamic Flow Service definition for UCA-EBC:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<adapter name="UCA-EBC" version="1.0"
xmlns="http://hp.com/umb/config">
  <flowServices>
    <flow name="DB-Flow" type="Dynamic"

collectorClass="com.hp.uca.expert.mediation.adapter.UcaDbCollector">
  <parameters>
    <parameter key="vp" defaultValue="" />
    <parameter key="notifier" defaultValue="dbNotifier" />
    <parameter key="summarize" defaultValue="false" />
    <parameter key="eligibilityScope" defaultValue="true"
/>  </parameters>
  </flow>
</flowServices>
</adapter>
```

### 3.8.2 UCA-EBC value pack configurations

### 3.8.3 Configuring a value pack for collecting events from an UMB flow

Each UCA-EBC value pack can be configured to collect events from UMB event flows provided by some distant UMB adapter Flow Providers.

Such configuration is made in the ValuePackConfiguration.xml file of the value pack in the 'mediationFlows' section.

An UMB consumer flow is defined with the Tag "UMBmediationFlow" has in the example below:

```
<mediationFlows>
  <UMBmediationFlow name="TeMIPflowOnOc1"
    targetAdapterName="TeMIP"
    targetFlowName="temipFlow1"
    automaticStart="true">
    <flowParameters>
      <flowParameter key="operationContext" value="OC1"/>
    </flowParameters>
  </UMBmediationFlow>
  <UMBmediationFlow name="SmartFlow"
    targetAdapterName="SMART"
    targetFlowName="smartFlow1"
    automaticStart="true"/>
</mediationFlows>
```

In this section, each UMBmediationFlow is defined specifying the following attributes:

- **name:** this is the local flow name.

- **targetAdapterName:** is the identifier of the adapter providing the production flow service
- **targetFlowName:** is the name of the flow definition on the target Adapter
- **automaticStart:** can be 'true' or 'false'. Indicates if the flow must be started along with the value pack. If omitted, the default value is 'true'. When set to false, the flow is not started at VP startup; it will have to be started manually from the GUI to become active.

Some flow creations require parameters to be provided (expected by the producer side). Flow parameters are defined in the `<flowParameters>` section of the `UMBmediationFlow`. Each parameter is a key/value pair defined with the `<flowParameter>` Tag with the following attributes:

- **key:** the parameter name
- **value** the parameter value

### 3.8.4 Forwarding Alarms to UMB through Static flows

One of the roles of the value packs is to forward correlation results (whatever their types: Alarms, Trouble tickets, events...) to some other applications.

From a scenario this is done by using an `UMBForwarder` object that makes the link between the scenario and the UCA-EBC flow service as defined in the `AdapterConfiguration.xml` file.

An `UMBEventForwarder` object can be easily created by requesting its creation from the value pack's Spring context (`context.xml` in the valuepack configuration directory).

Here is an example of `UMBEventForwarder` creation:

```
<bean name="mediationEventForwarder"
class="com.hp.uca.expert.event.UMBEventForwarder">
  <constructor-arg index="0">
    <value>UcaStaticEventForwarderFlow</value>
  </constructor-arg>
</bean>
```

The `UMBEventForwarder` object is created with an argument which is the name of the static flow as it is define in the UCA-EBC `AdapterConfiguration.xml` file.

Then from a rule file, this `UMBEventForwarder` object can be used as follow:

1. Define the object in the rule file 'global section'
2. Use the `UMBEventForwarder push()` method to forward an event to the bus.

Example of rule forwarding an event to the bus:

```

package com.hp.uca.expert.vp.alarmforwarder;

#list any import classes here.
import com.hp.uca.expert.event.EventForwarder;
import com.hp.uca.expert.event.Event;
import com.hp.uca.expert.x733alarm.PerceivedSeverity;
import com.hp.uca.expert.util.MessageFileHandler;
import java.util.ArrayList;
import com.hp.uca.expert.scenario.Scenario;
import com.hp.uca.common.trace.LogHelper;
import com.hp.uca.expert.flag.Flag;
import
com.hp.uca.expert.testmaterial.AbstractJUnitIntegrationTest;

#declare any global variables here
global Scenario theScenario;
global EventForwarder mediationEventForwarder;

# Forward any event received
rule "Forward any event received"
no-loop
    when
        $event : Event()
    then
        LogHelper.enter(theScenario.getLogger(),
drools.getRule().getName());

        // Forward the event to ne new Mediation
mediationEventForwarder.push($event);

        // Retract the event
        theScenario.getLogger().info("Retracting: \n"+
$event.toFormattedString());
        theScenario.getSession().retract($event);

        LogHelper.exit(theScenario.getLogger(),
drools.getRule().getName());
    end

```

### 3.8.5 Forwarding Alarms to UMB through Dynamic flows

Starting V3.3, it is possible to use a UCA to UMB dynamic flow to retrieve alarms stored in a Database.

Please refer to the “UCA for EBC Reference Guide”

# Unified Mediation Bus Adapter Development Kit

The Unified Mediation Bus Adapter Development Kit is running and supported on Windows and Linux. It is delivered as follows:

**On Windows XP/Vista 64 bits, Windows 7 64 bits, Windows Server 2012:**

`umb-adapter-dev-package-1.0-msi.zip`

**On Linux:**

`umb-adapter-dev-package-1.0-linux.tar`

This chapter describes the software prerequisites, the installation steps, and gives a brief content description of the UMB Adapter Development kit.

## 4.1 Licensing

Please refer to Chapter 2.1 “*Licensing*” for more information on UMB Development Kit licensing.

## 4.2 Disk requirements

Here are the disk requirements for the UMB Development Kit:

Type	Disk requirements
Installation time temporary disk space	40 MB minimum: <ul style="list-style-type: none"><li>• 20 MB minimum for the <code>umb-adapter-dev-package-1.0</code> archive file</li><li>• 20 MB minimum for files expanded from the <code>umb-adapter-dev-package-1.0</code> archive file)</li></ul>
Permanent disk space	20 MB minimum for UMB Development Kit V1.0 installed on the system

**Table 12 - Disk Requirements for UMB Development kit**

## 4.3 Software prerequisites

### 4.3.1 Java

UMB V1.0 Adapter Development Toolkit requires Java JDK 1.7.



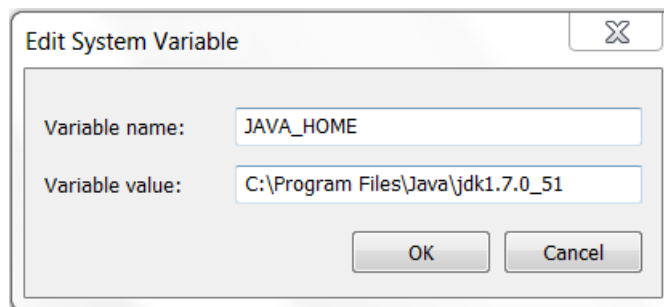
Software	Version
Java JDK 7	1.7.0.00 (or later)

**Table 13 - Software Prerequisites for UMB Adapter Development Kit**

The JAVA\_HOME environment variable must be set before using UMB Development Kit:

#### On Windows:

In the *Control Panel*, Open *System Properties*, open the *Advanced* tab and click *Environment Variables*, then set the JAVA\_HOME environment variable according to the location of your JDK:



**Figure 17 - Setting the JAVA\_HOME environment variable on Windows systems**

In case Java is not yet installed on your system, the latest JDK package for Microsoft Windows operating systems can be downloaded (for free) from <http://java.com/en/download/manual.jsp>.

#### On Linux:

Depending on your shell, and the location of the Java JDK software, please use one of the following commands to set the JAVA\_HOME environment variable:

Example for **csh**-like shell:

```
$ setenv JAVA_HOME /opt/java/jdk1.7.0_51
```

Example for **sh**-like shell:

```
$ export JAVA_HOME=/opt/java/jdk1.7.0_51
```

To check if you already have Java installed:

```
$ rpm -qa | grep jdk
```

Red Hat Enterprise Linux Server comes with OpenJDK Java VM. You should get an output similar to the following:

```
java-1.7.0-openjdk-1.7.0.9-2.3.4.1.el6_3.x86_64  
java-1.7.0-openjdk-devel-1.7.0.9-2.3.4.1.el6_3.x86_64
```

You can also download (for free) the latest Java packages (HotSpot Java VM) from Oracle from <http://java.com/en/download/manual.jsp>. If this is installed (usually under `/usr/java`), you should get an output similar to the following:

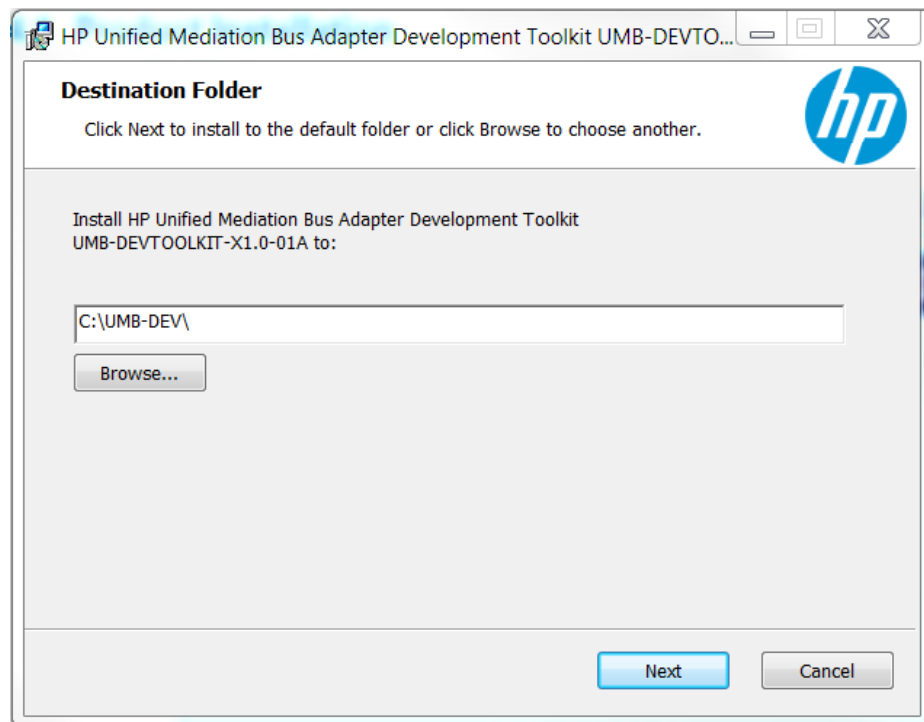
```
jdk-1.7.0_51-fcs.x86_64
```

## 4.4 Unified Mediation Bus Adapter Development Kit installation

### 4.4.1 Product Installation

#### On Windows:

Install the UMB Adapter Development Kit by executing the UMB-DEVTOOLKIT-V1.0-0A.msi file.



**Figure 18 - Installing UMB Adapter Development Kit**

By default, the UMB Adapter Development Kit is installed in the `C : \UMB-DEV` directory.

The installer automatically creates/updates some environment variables such as:

- The system's PATH environment variable is updated in order to make 3rd party product executables (i.e. Apache Ant) easily available
- The UMB\_DEV\_HOME environment variable that stores the UMB Adapter Development Kit root directory (by default `C : \UMB-DEV`)

---

### Note

---

On Windows, you must open a new CMD.EXE window in order to benefit from the new/updated environment variables.

---

#### On Linux:

- Un-tar the archive in a temporary directory:

As **root** user, un-tar the archive in a temporary local directory (For example: /tmp):

```
$ cd /tmp
$ tar -xvf <kit location>/umb-adapter-dev-package-1.0-
linux.tar
```

- Run the installation script

Depending on whether you wish to install the UMB Adapter Development Kit at the default location, i.e. /opt/UMB-DEV, or in an alternate location, run either of the following commands to execute the installation script.

To install UMB Adapter Development Kit at the default location (in /opt/UMB-DEV directory), please execute the following command as **root** user:

```
$ install-umb-dev.sh
```

To install the UMB Adapter Development Kit at an alternate location of your choosing, please execute the following command as **root** user:

```
$ install-umb-dev.sh -r <Alternate root directory>
```

- Post-installation setup : setting the environment variables:

The UMB Adapter Development Kit on Linux requires the UMB\_DEV\_HOME environment variable to be set in order to work properly.

For that purpose, the UMB Adapter Development Kit installation script installs two files in the UMB Adapter Development Kit root directory:

By default:

- /opt/UMB-DEV/.adapterdev\_environment.sh
- /opt/UMB-DEV/.adapterdev\_environment.csh

These files can be used for setting the correct environment variables for the user account(s) that will be using the UMB Adapter Development Kit.

Depending on your shell, use one of the following commands to set the UMB Adapter Development Kit environment variables and update the path:

On **csh**-like shell:

```
$ source /opt/UMB-DEV/.adapterdev_environment.csh
```

On **sh**-like shell:

```
$ . /opt/UMB-DEV/.adapterdev_environment.sh
```

---

## Note

---

### Installing UMB Adapter Development kit as non-root user (Linux only):

For testing purpose (or for some very specific needs) the UMB Adapter Development Kit package can be installed by a non-root user. This feature is available for Linux only.

When installing UMB Adapter Development Kit as non-root user, the following limitations must be understood and acknowledged:

1. The system RPM database is not accessible by a non-root user. As a consequence, when installation is performed by a non-root user, a specific RPM database must be specified. The default RPM repository for non-root installation is set to `~/ .rpmdb` (where `~` is the user home directory). This directory can be overridden by specifying the `--rpmdbpath` option as installation script argument.
  2. The UMB Development Kit root directory must be read/write accessible by the non-root user. Usually the default `/opt/UMB-DEV` directory cannot be used (unless some specific rights have been set by the administrator). As a consequence, when installation is performed by a non-root user, the `-r` option **must be** specified.
  3. When installed by the non-root users the UMB Development Kit files are owned by the user who performed the installation.
- 

## 4.4.2 Files organization

The UMB Adapter Development Kit is installed under the `%UMB_DEV_HOME%` directory on Windows or the `${UMB_DEV_HOME}` directory on Linux, which is by default the `C:\UMB-DEV` directory on Windows or the `/opt/UMB-DEV` directory on Linux.

The following table describes the different subdirectories.

Directories	Description
<i>3pp</i>	Contains the third party tools delivered with the Adapter Development Toolkit (mainly ant)
<i>adapter-examples</i>	Contains a set of Adapter examples used to demonstrate the UMB capability in different domains.
<i>apidoc</i>	Contains the Javadoc of the Java classes provided by UMB that can be used in adapter development.
<i>bin</i>	Contains the un-installer tool
<i>eclipseplugin</i>	Contains the eclipse plugin and associated template files
<i>lib</i>	Contains the jar files required by the developed adapters.

**Table 14 - Sub-directories of UMB Adapter Development Kit installation directory**

### 4.4.3 Setting the Unified Mediation Bus Adapter Development Toolkit environment variables (Linux only)

Several environment variables must be defined for UMB Adapter Development Toolkit to work properly.

For that purpose, the UMB Adapter Development Toolkit installation script installs two files in the UMB HOME directory (*/opt/UMB-DEV* by default):

- *.adapterdev\_environment.sh*
- *.adapterdev\_environment.csh*

These files can be used for setting the correct environment variables.

Depending on your shell, use one of the following commands to set the “hpossadm” user’s UMB environment variables and update the path:

On **csh**-like shell:

```
$ source /opt/UMB-DEV/.adapterdev_environment.csh
```

On **sh**-like shell:

```
$ . /opt/UMB-DEV/.adapterdev_environment.sh
```

## 4.5 Un-installation of UMB Adapter Development Kit

In order to uninstall the UMB Adapter Development Kit, please follow the instructions below:

**On Windows:**

5. Go to the **Control Panel**
6. Select **“Program and Features”**
7. Right-click on **“HP Unified Mediation Bus Development toolkit – UMB-DEVTOLKIT-V1.0-0A”**
8. Select **“Uninstall”**

**On Linux:**

```
$ /opt/UMB-DEV/bin/uninstall.sh
```

You should get an output similar to the following text:

```
Here is the list of installed UMB-DEV packages:
```

```
[0]      UMB-DEVTOLKIT-V1.0-0A
```

```
Enter the index number of UMB-DEV version to un-install:
```

By entering ‘0’ (as in the example above), UMB Development Toolkit version V1.0-0A will be removed.

## Code Signing

This Software Product from HP is digitally signed and accompanied by Gnu Privacy Guard (GnuPG) key.

### 5.1 On Red Hat Enterprise Linux and HP-UX platforms

Below mentioned procedure\* allows you to assess the integrity of the delivered Product before installing it, by verifying the signature of the software packages.

Pick the signature (.sig) file shipped along with the product and use following GPG command

```
gpg --verify <product.sig> <product>
```

Example: `gpg --verify VPNSVP-X51-3A.zip.sig VPNSVP-X51-3A.zip`

## Note: Look for the comments shown below in the command output

**Good signature from "Hewlett-Packard Company (HP Code signing Service)"**

=====

Note: If you are not familiar with signature verification using GPG and intended to verify HP Product signature, follow the steps given below.

1. Check whether gnupg gpg is installed on the system. If no, install gnupg gpg
2. Configure GPG for accepting HP signature. The steps are the following:
  - a. Log as root on your system
  - b. Get the hpPublicKey from following location:  
<https://h20392.www2.hp.com/portal/swdepot/displayProductInfo.do?productNumber=HPLinuxCodeSigning> and save it as hpPublicKey.pub  
Note that the hpPublicKey file will be located in the root's home directory.
  - c. Follow the instruction found at above URL in the "Verification using GPG" section.

*\*HP strongly recommends using signature verification on its products, but there is no obligation. Customers will have the choice of running this verification or not as per their IT Policies.*

---

# Glossary

UCA: Unified Correlation Analyzer

UMB: Unified Mediation Bus

EBC: Event Based Correlation

JDK: Java Development Kit

JMS: Java Messaging Service

JMX: Java Management eXtension, used to access or process action on the UMB product

JNDI: Java Naming and Directory Interface

JRE: Java Runtime Environment

Inference Engine: Process that uses a Rete algorithm

DRL: Drools Rule file

XML: Extensible Markup Language

XSD: Schema of an XML file, describing its structure

X733: Standard describing the structure of an Alarm used in telecommunication environment