Unified Mediation Bus Release notes

Version 1.1
Edition: 1.0

Hewlett Packard Enterprise

Notices

Legal notice

The information contained herein is subject to change without notice. The only warranties for HPE 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. HPE shall not be liable for technical or editorial errors or omissions contained herein.

Printed in the US

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.

Trademarks

Adobe®, Acrobat® and PostScript® are trademarks of Adobe Systems Incorporated.
Java™ is a trademark of Oracle and/or its affiliates.
Microsoft®, Internet Explorer, Windows®, Windows Server®, and Windows NT® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Firefox® is a registered trademark of the Mozilla Foundation.
Google Chrome® is a trademark of Google Inc.
Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.
UNIX® is a registered trademark of The Open Group.
X/Open® is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.
Red Hat® is a registered trademark of the Red Hat Company.
Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.
Neo4j is a trademark of Neo Technology.
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.

Table of Contents

Figures

List of tables

Preface

About this guide

These Release Notes describe critical information related to the HP Unified Mediation Bus product.

Product Name: Unified Mediation Bus

Product Version: 1.1

Kit Version: V1.1

Please read this document before installing or using this Software.

Intended Audience

Here are some recommendations based on possible reader profiles:

- Solution Developers
- Solution Integrators

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
Unified Mediation Bus Server Version 1.1	• Red Hat Enterprise Linux Server release RHEL 6.x & 7.x
Unified Mediation Bus Runtime Version 1.1	• Red Hat Enterprise Linux Server release RHEL 6.x & 7.x • Windows XP / Vista 64 bits • Windows Server 2012 • Windows 764 bits
Unified Mediation Bus Adapters Development Toolkit Version 1.1	• Red Hat Enterprise Linux Server release RHEL 6.x & 7.x • Windows XP / Vista 64 bits • Windows Server 2012 • Windows 764 bits
Unified Mediation Bus TeMIP Adapter Version 1.1	• HP-UX 11.31 for Itanium • Red Hat Enterprise Linux Server release RHEL 6.x & 7.x
Unified Mediation Bus Exec Adapter Version 1.1	• HP-UX 11.31 for Itanium • Red Hat Enterprise Linux Server release RHEL 6.x & 7.x • Windows XP / Vista 64 bits • Windows Server 2012

Product Version	Supported Operating systems
	• Windows 7
Unified Mediation Bus OSS Analytics Foundation Adapter Version 1.1	• HP-UX 11.31 for Itanium • Red Hat Enterprise Linux Server release RHEL 6.x & 7.x • Windows XP / Vista 64 bits • Windows Server 2012 • Windows 7
Unified Mediation Bus Camel Adapter Version 1.1	• Red Hat Enterprise Linux Server release RHEL 6.x & 7.x • 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

HP Unified Mediation Bus - Installation and Configuration Guide

HP Unified Mediation Bus - Adapter Development Guide

Support

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

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

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

Chapter 1
Changed and new features

Unified Mediation Bus (UMB) V1.1 is the OSS mediation product based on recent technologies from the open-source community like Kafka/Zookeeper and Hazelcast. It addresses:

- operations driven use cases that could not be addressed with the current technology based on ServiceMix, ActiveMQ, Camel and JBI
- use cases needing extremely high throughput
- use cases needing extremely high availability
- configuration tasks in highly distributed solutions

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

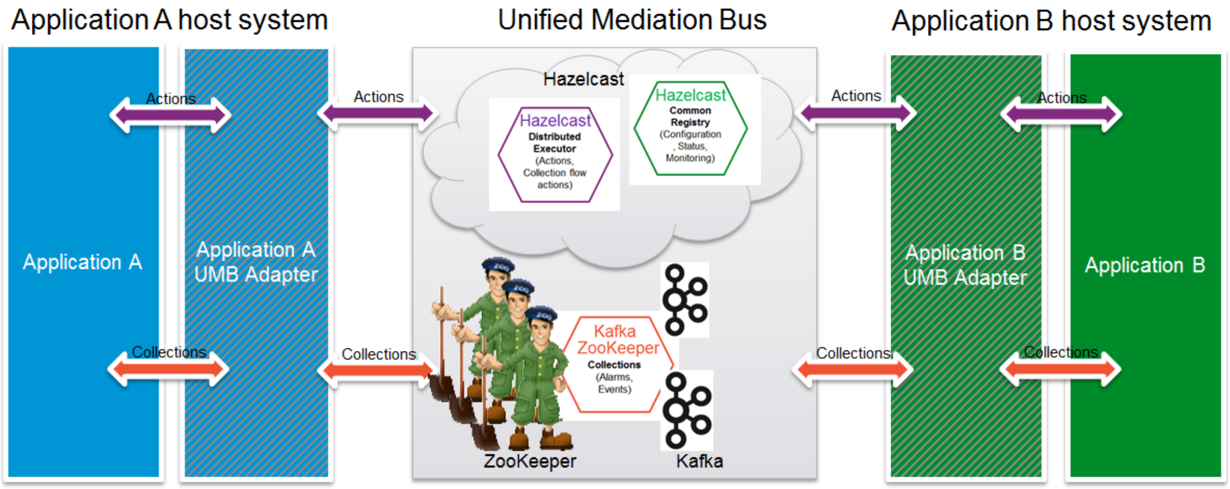


Figure 1 - Unified Mediation Bus architecture overview

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

1.1 Main features provided by this release

The following features have been implemented for version 1.1

1.1.1 UMB server

- The UMB server package is now based on Apache Kafka 0.9.0.1. This is the latest available stable version of Apache Kafka.
- Zookeeper & Kafka as a Linux service is now supported on RHEL7 where default startup service is based on systemd

1.1.2 UMB framework

1.1.2.1 Introduction of UMB Clusters

A UMB cluster is a group of UMB adapters than can work together, or in other words, a set of adapters that can communicate and share information with each other.

To be part of the same Cluster, adapters must share two things:

- Must belong to the same Hazelcast Group
- Must share the same network configuration

By default, at installation, a UMB Cluster with the name `default` is created. The location of the defined clusters is under the directory `${UMB_DATA}/clusters`.

The `default` cluster is therefore defined in: `${UMB_DATA}/clusters/default`.

Each cluster is configured by the `cluster-config.xml` file in its `conf` directory.

Please refer to the *UMB installation, configuration and Administration Guide* for a full description of this feature.

1.1.2.2 New `umb` Administration command

A '`umb`' command is now provided to administrate UMB:

- Deploy adapters to a UMB Cluster
- Start / stop Adapters
- Show Adapter running on a UMB Cluster
- Get the status of adapters (local or Cluster wide)
- Get the description of established flows

Please refer to the *UMB installation, configuration and Administration Guide* for a full description of this feature.

1.1.2.3 Adapter Name is now optional if action Group is defined

The goal of this feature is to be able to not specify any name for an UMB Adapter in the case that there is an `actionGroup` defined in the `AdapterConfiguration.xml` file. A name will be generated automatically at the start of the Adapter based on the name of the `actionGroup` if no name is provided for the Adapter. The format for automatically generated Adapter names is: *`<actionGroup>-<adapter number>`*. For example, if the actionGroup name is `myActionGroup` then adapter names will be: `myActionGroup-1`, `myActionGroup-2`, etc…

As a reminder, `actionGroups` are used to provide load-balancing across a set of adapters that provide the same action services.

1.1.2.4 Possibility to define a UMB broker on a per flow basis

The goal of this feature is to allow flows to produce events to several Kafka/Zookeeper clusters.

The feature requires setting the new `broker` attribute in the flow definition. The `broker` attribute is to be set in the `AdapterConfiguration.xml` file for each flow that wishes to produce to a specific Kafka/Zookeeper clusters. The `broker` attribute indicates the Kafka/Zookeeper broker cluster to be used by the flow for producing events.

The broker attribute value refers to a set of properties (prefixed by the broker attribute value) in the `adapter.properties` file which now allows Kafka/Zookeeper cluster specific settings.

Refer to "*UMB Adapter Development Guide*: Multiple Kafka/Zookeeper clusters configuration" for full description of this feature.

1.1.2.5 Possibility to set the read offset (earliest or latest) when starting a consumer

This feature allows to set the read offset ('`earliest`' or '`latest`') when starting a consumer. The read offset determines the way in which a consumer consumes messages from a Kafka broker.

The read offset '`earliest`' value corresponds to the consumption of all messages in the Kafka server while '`latest`' indicates the consumption of only messages that are sent from the moment the consumer is started onwards.

The feature requires the setting of a new attribute (readOffset) in the `AdapterConfiguration.xml` file of the consumer. Such attribute can be set to '`earliest`' or '`latest`'.

The readOffset attribute is optional but when set, it overrides the value of the consumer.auto.offset.reset property specified in the `adapter.properties` file of the same consumer. If the readOffset attribute is not specified, the read offset value will be equal to the value of the consumer.auto.offset.reset property or equal to the default value of '`latest`' if the consumer.auto.offset.reset property is not set.

Refer to "*UMB Adapter Development Guide*: Defining Flow Service Consumers from Configuration" for full description of this feature.

1.1.2.6 Possibility to specify a list of values in Action parameter definition

There is now the possibility to specify a list of possible values for an Action parameter. This is done by setting the ‘`listValues`’ optional attribute with a list of comma separated values.

1.1.2.7 High Availability: notion of Standby Adapters

To respond to the High Availability requirement for Adapters, UMB V1.1 brings the new notion of “Standby Adapters”.

A Standby Adapter is an adapter that has the exact same configuration as another existing adapter:

- Same name
- Same service definitions

A Standby Adapter is inactive as long as the corresponding Master Adapter is in ‘Running’ State. Only one standby adapter is allowed.

When an Adapter is started twice, the second instance of the adapter will be in ‘Standby’ State.

When a ‘Running’ Adapter is stopped and if a ‘Standby’ Adapter exists in the cluster for this Adapter, then the ‘Standby’ Adapter will switch to ‘Running’ State as soon as it detects that the ‘Running’ adapter has stopped. The formerly “Standby’ adapter becomes active and provides the same flow and action services as the now stopped adapter.

1.1.2.8 Possibility to specify a Topic Name on the producer side for static flows

When a flow Consumer is not a UMB Adapter, it is necessary to set a fixed name for the Flow Topic on the producer side in order for the consumer to be properly configured.

Example of flow definition specifying a topic name:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<adapter name="ProducerAdapter" version="1.0"
xmlns="http://hp.com/umb/config">
    <flowServices>
        <flow name="StaticFlow1" type="Static"
topicName="TopicName1"

collectorClass="com.hp.umb.adapter.explicittopic.ExplicitTopicCol
lector">
        </flow>
  </flowServices>
</adapter>
```

1.1.2.9 Possibility to specify the number of partitions on the producer side for static flows

The number of partitions can now be specified for a flow topic on the producer side.

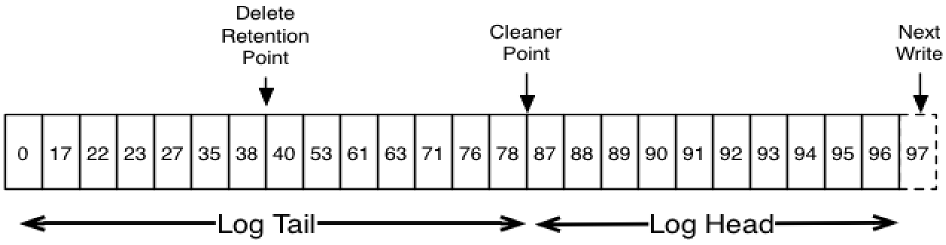
Example of flow definition specifying the number of partitions:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<adapter name="ProducerAdapter" version="1.0"
xmlns="http://hp.com/umb/config">
    <flowServices>
        <flow name="StaticFlow1" type="Static"
numberOfPartitions="3"

collectorClass="com.hp.umb.adapter.explicittopic.ExplicitTopicCol
lector">
        </flow>
  </flowServices>
</adapter>
```

1.1.2.10 Possibility to specify the “log compaction” policy per flow:

To save disk space without losing all old messages Kafka provides the option to retain them even if they are behind the cleaner point.



The old implementation was simply deleting old segments of log, now it is possible to save all old messages that have a different key.

Example: the topic generated by this flow will use log compaction.

```
<flowServices>
  <flow name="StaticFlow" type="Static" logCleanupPolicy="compact" …>
     </flow>
</flowServices>
```

1.1.2.11 Replication factor per flow

At the moment of creation of a new topic it is necessary to specify the number of servers where every message will be replicated (for failover purposes) or the default value ‘1’ will be used (no replication).

Example: the topic generated by this flow will be replicated on two servers.

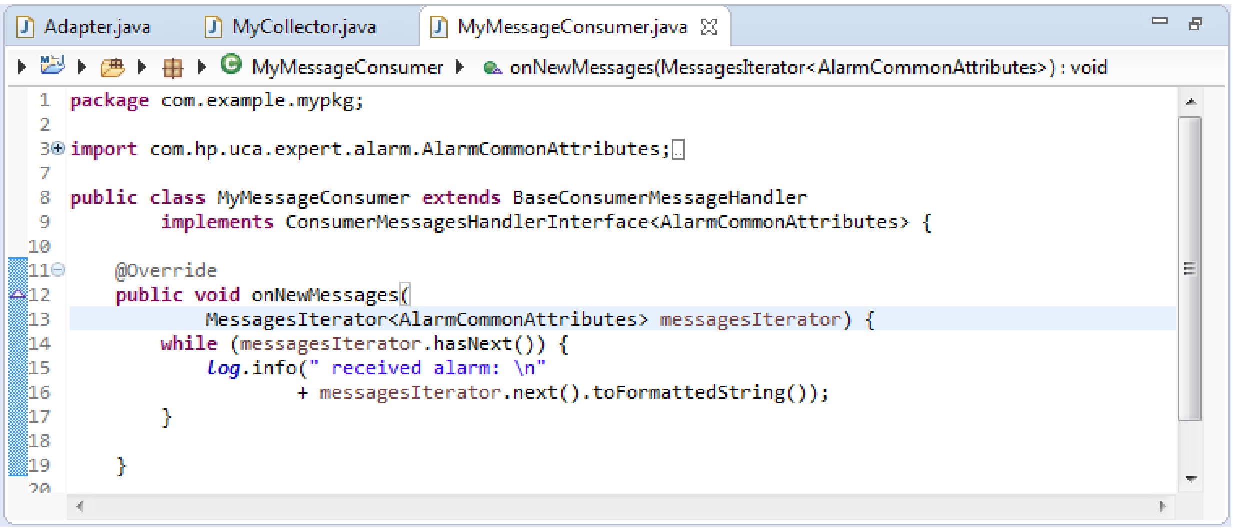
```
<flowServices>
  <flow name="StaticFlow" type="Static" replicationFactor="2" …>
     </flow>
</flowServices>
```

If the replication factor is higher than the number of existing brokers then an exception like the following is thrown:

```
Exception in thread "main" kafka.admin.AdminOperationException: replication factor: 2
larger than available brokers: 1
```

1.1.2.12 New ConsumerMessageHandlerGenericInterface<K>

Since UMB V1.1 the ConsumerMessageHandlerGenericInterface<K> is the recommended interface for implementing a message consumer.
A typical message consumer class should now be similar to the following:



```
package com.example.mypkg;

import com.hp.uca.expert.alarm.AlarmCommonAttributes;

public class MyMessageConsumer extends BaseConsumerMessageHandler
        implements ConsumerMessagesHandlerInterface<AlarmCommonAttributes> {

    @Override
    public void onNewMessages(
            MessagesIterator<AlarmCommonAttributes> messagesIterator) {
        while (messagesIterator.hasNext()) {
            log.info(" received alarm: \n"
                    + messagesIterator.next().toFormattedString());
        }

    }
```

1.1.2.13 Manual commit of consumed messages

The ‘automatic commit’ of consumed messages can be de-activated letting the Consumer Class the possibility to manually commit the messages.

This new feature is detailed in the “UMB Adapter Development Guide” chapter: “Use the Manual Commit feature”.

1.1.3 UMB Adapter Development Kit

1.1.3.1 (HPE private only) UMB framework dependencies can be specified as BOM in maven builds.

NOTICE: This is for an internal HPE Use.

When building an Adapter with Maven, the UMB BOM (Bill of materials) can be used as dependency.
This allows for a simpler dependency management.

To use the UMB BOM, the following must be specified in the "Managed Dependencies" section of the `pom.xml` file:

```
<dependencyManagement>
    <dependencies>
          <dependency>
              <groupId>com.hp.umb</groupId>
              <artifactId>umb-bom</artifactId>
              <version>1.1</version>
              <type>pom</type>
              <scope>import</scope>
          </dependency>
    </dependencies>
</dependencyManagement>
```

Then it is not necessary anymore to specify the versions for the dependencies that are provided by the UMB BOM file:

```
<dependencies>
    <dependency>
        <groupId>com.hp.umb</groupId>
        <artifactId>umb-adapter-fmk</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>provided</scope>
    </dependency>
```

1.1.3.2 The UMB Development environment is now fully based on Maven (instead of Ant)

All the UMB provided examples are now delivered with a POM file that allows the user to build them using the Apache Maven tool.

The UMB Adapter development toolkit comes with a Maven repository containing all the UMB framework libraries and their dependencies.

The Apache Maven tool is not provided in the UMB Development Kit and it is the responsibility of the UMB developer to download, install and configure it. UMB V1.1 Adapter Development Toolkit requires version 3.3 of Apache Maven at the minimum. This tool can be downloaded at https://maven.apache.org/

1.1.3.3 The UMB eclipse plugin can now generate metric definition projects

Some adapters requires the development of specific Event classes. The project skeletons for developing such classes can now be produced by the UMB Eclipse plug-in.

1.1.3.4 The UMB eclipse plugin has been reworked to provide maven compliant projects

The Eclipse plug-in now produces adapter and metric definition projects that can be built using Apache Maven.

1.1.3.5 Migration to Kafka version 0.9.0.1

- With the new Kafka implementation some of the most important properties names and values have changed:

  auto.offset.reset=smallest/largest → auto.offset.reset=earliest/latest

  metadata.broker.list → bootstrap.servers

- The `adapter.properties` file must contain the `consumer.bootstrap.servers` property when it is not specified by the producer in the same file.

- It is possible to specify a key and/or value serializer for the producer and a key and value de-serializer for the consumer. In fact those properties are mandatory for with Kafka 0.9.0.1 but the UMB framework replaces them with a default serializer when they are missing.

  Kafka provides built-in serializers for common objects (see Kafka 0.9.x docs):

  *producer.key.serializer=org.apache.kafka.common.serialization.**StringSerializer***
  *producer.value.serializer=org.apache.kafka.common.serialization.**StringSerializer***

  *consumer.key.deserializer=org.apache.kafka.common.serialization.**StringDeserializer***
  *consumer.value.deserializer=org.apache.kafka.common.serialization.**StringDeserializer***

But it is also possible to specify custom ones:

producer.key.serializer=com.package.MySerializer

A custom Serializer/Deserializer class must comply with the following:

- Have a no-arguments constructor
- Extend *com.hp.umb.adapter.internal.utilities.JavaClassSerializer* OR implement *org.apache.kafka.common.serialization.Serializer<Object>*, *org.apache.kafka.common.serialization.Deserializer<Object>*

NOTE:
For backward compatibility in the tests, there are still custom adapters that implement old interfaces. Those will be completely replaced by new versions in the future.

All obsolete properties have been removed from the `adapter.properties` files in the UMB Framework and all the example adapters.

NOTE:
The `zookeeper.connect` property is not anymore a property either of the producer nor the consumer. It became a general property that is **mandatory for the producer adapters** but that can be retrieved from the producer flow (when possible) by the consumer adapters. When it is not possible to get the property from the producer flow then it must be present in the consumer adapter properties file.

1.1.4 Interoperability with UMB V1.0

UMB V1.1 can interoperate with UMB V1.0, provided that:

- UMB V1.1 adapters use UMB V1.1 Runtime
- UMB V1.0 adapters use UMB V1.0 Runtime Patch 3 (UMB10RTLIN_00003 on Linux, UMB10RTHPUX_00003 on HP-UX or UMB10RTWIN_00003 on Windows) or above.

NOTE:
When connecting UMB V1.1 consumer adapters to UMB V1.0 producer adapters, it is mandatory to define the `producer.bootstrap.servers` property in the `adapter.properties` file of consumer adapters because it is not provided by the UMB V1.0 producer adapter definition.

The value of the `producer.bootstrap.servers` property is a comma-separated list of Kafka broker `<host>:<port>` information. For example:

```
producer.bootstrap.servers=host1:9092,host2:9092
```

Please see chapter 1.1.3.5 “Migration to Kafka version 0.9.0.1” for more information on property changes from UMB V1.0 (that uses Kafka 0.8.2.0) to UMBV1.1 (that uses Kafka 0.9.0.1).

1.1.5 UMB documentation

1.1.5.1 Documentation reworked to apply new HPE Template

The documentation has been fully reworked to comply with the new HPE template.

1.1.5.2 Documentation enhancements

The documentation has been enriched with a better description of some UMB framework mechanisms.

A new chapter has been added to the *UMB Adapter Development Guide* that describes the concurrency, thread safety and lifecycle of user-provided classes. Indeed, the UMB Framework requires that the user provides Java classes that give the implementation for specific features of an Adapter. This is the case for:

- Action services: the action service provider Adapter developer is required to provide the name of a Java class that provides the implementation of each action service. The name of the class is defined in the **`actionClass`** attribute of the `action` XML element defining the action service in the `AdapterConfiguration.xml` file
- Flow services: the flow producer Adapter developer is required to provide the name of a Java class that provides the implementation of each flow service. The name of the class is defined in the **`collectorClass`** attribute of the `flow` XML element defining the flow service in the `AdapterConfiguration.xml` file. Optionally, it is also possible to provide the name of a Java class to serialize/de-serialize flow messages, in case the default serialization class

(`com.hp.umb.adapter.internal.utilities.JavaClassSerializer`) is not wanted. The name of this class is defined in the **`serializerClass`** attribute of the `flow` XML element defining the flow service in the `AdapterConfiguration.xml` file.

- Automatic flow consumers: Both standard (UMB) and non UMB flows can have automatic flow consumers. The automatic flow consumers are defined in the `AdapterConfiguration.xml` file of the flow consumer Adapter, using either the `autoConsumer` or the `autoNonUMBConsumer` XML element. In order to define an automatic flow consumer, the developer is required to provide the name of the Java class that provides the implementation of the consumer. The name of the class is defined in the **`messageConsumerClass`** attribute of the `autoConsumer` or `autoNonUMBConsumer` XML element defining the automatic consumer in the `AdapterConfiguration.xml` file. Optionally, it is also possible to provide the name of a Java class to serialize/de-serialize flow messages, in case the default serialization class (`com.hp.umb.adapter.internal.utilities.JavaClassSerializer`) is not wanted. The name of this class is defined in the **`serializerClass`** attribute of the `autoConsumer` or `autoNonUMBConsumer` XML element defining the automatic consumer in the `AdapterConfiguration.xml` file.

The new chapter describes the life-cycle and thread-safety of the **`actionClass`**, **`collectorClass`**, **`messageConsumerClass`** and **`serializerClass`** objects.

Some documentation defects have been fixed. See the “Fixed Problems” section for a full description of the problem fixed.

Chapter 2
Fixed Problems

The following problems/enhancements were fixed/implemented in this release

Reference Severity	Component	Description	Comment
CR#14153 Medium	Documentation	Admin & Config guide should take into account firewalld (new in rhel7)	Fixed
CR#14148 Medium	Development Kit	UMB Adapter Toolkit Location not recognized as valid in UMB Eclipse Plugin	Fixed
CR#14146 Medium	Development Kit	umb-example-metrics in UMB V1.1 Dev Kit fails to build (JAXB issue)	Fixed
CR#14144 Medium	Framework	Clash of commons-logging and jcl-over-slf4j	Fixed
CR#14079 High	Framework	UMB: A mechanism should be put in place to limit the number of retries for a monitored connection	Fixed
CR#14076 Urgent	Framework	OutOfMemory Exception when a monitored flow fails at startup	Fixed
CR#14031 High	Framework	UMB: need a way to force the Topic name and number of partitions for a static flow	Enhancement implemented
CR#14003 Medium	Packaging	kafka-linux installation fails if missing 'ksh'	Fixed
CR#13675 Medium	Framework	High availability for flow services	Enhancement implemented
CR#13492 Medium	Framework	Kafka server fails to start again if it was stopped when a client was connected to it	Fixed

Table 2 - Fixed Problems

Chapter 3
Known Problems

This section lists problems discovered during the product test campaign or previous releases, and that still have to be fixed.

Reference / Severity	Problem Description
CR#14603 Low	Description: UMB command reports wrong message when no adapter is running Status: This message can be ignored. The UMB command will be fixed in a future release.
CR#14479 Low	Description: Impossible to activate hazelcast traces for UMB command Status: Will be investigated for a future version
CR#14477 Low	Description: On Linux the "umb -v start adapterName" does not work as expected Status: Will be investigated for a future version
CR#14468 Medium	Description: Cannot install patch UMB10RTLIN_00003 in non-root environment Status: Will be investigated for a future version
CR#14005 Low	Description: zookeeper and kafka do not use JAVA_HOME set in environment Status: Will be investigated for a future version
CR#14002 Medium	Description: Bad message “The user hpossadm has been automatically created during installation. Please assign this user a password in order to be able to login.” displayed on zookeeper and kafka installation Status: This message can be ignored. The script will be fixed in a future release.
CR#13866 Low	Description: Interrupted exception in com.hp.uca.common.queue.MonitoredQueue.publish while VP is restarting Status: Will be investigated for a future version
CR#13856 Medium	Description: Flow API sometimes breaks events order Workaround: When implementing the collector for a new adapter, please implement the Collection<Event> pull() method using a java.util.List instead of a java.util.Collection as the return type of the method. This will preserve the order.
CR#13608 Medium	Description: UMB chkconfig --add kafka does not work Status: Will be investigated for a future version

Table 3 - Known Problems