# OSS Fault Analytics and Statistics

## Installation, Configuration and Administration Guide

Version 1.2.0

Edition 1.0 - December 2016

Hewlett Packard Enterprise

# Notices

### Legal notice



Printed in the US

### Trademarks

# Contents

# List of tables

# List of figures

# Preface

## About this guide

This guide describes how to install, configure, administrate and troubleshoot the HPE OSS Fault Analytics and Statistics product.

Product name: HPE OSS Fault Analytics and Statistics
Software component version: 1.2
Kit name: `ossa-fault-1.2-MR.noarch.rpm`

## Audience

This installation and configuration guide is for anyone who is responsible for installing, uninstalling, configuring, administrating, or troubleshooting the HPE OSS Fault Analytics and Statistics.

## Software versions

**The terms Unix and Linux are used as a generic reference to the operating system, unless otherwise specified. The software versions referred to in this document are as follows:**

**Table 1: Software versions**

| Product version | Supported operating systems |
|---|---|
| HPE OSS Analytics Foundation version 1.1.4 | Red Hat Enterprise Linux Server release 6.8 |
| HPE OSS Fault Analytics and Statistics version 1.2 | Red Hat Enterprise Linux Server release 6.8 |
| HPE Vertica version 7.2.3 | Red Hat Enterprise Linux Server release 6.8 |
| HPE UMB Server version 1.1 | Red Hat Enterprise Linux Server release 6.8 |
| HPE Unified OSS Console 2.3 | Red Hat Enterprise Linux Server release 6.8 |
| HPE TeMIP 6.2 | Red Hat Enterprise Linux Server release 6.8 |

## 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:

*HPE OSS Analytics Foundation Release Notes*
*HPE OSS Analytics Foundation Integration Guide*

*HPE OSS Fault Analytics and Statistics User Guide*
*HPE OSS Fault Analytics and Statistics Customization Guide*
*HPE TeMIP Analytics patch readme documents*
*Kafka documentation:* http://kafka.apache.org/documentation.html

# 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 web site includes the following:

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

# Chapter 1
# Product overview

## 1.1 Introduction

HPE OSS Fault Analytics and Statistics is a software product that enables telecommunications service providers with the capabilities to collect and persist fault information from fault and surveillance systems, transform the data as needed and deliver actionable insight to operations staff to operate and manage their network. The actionable insight is inferred using a host of statistical and analytical techniques.

OSS Fault Analytics and Statistics (FAS) is positioned as an independent product, working with fault information consolidated in HPE TeMIP, as well as any other surveillance system from an independent software vendor.

FAS is based on HPE Vertica, complemented by a mediation layer allowing for collection of fault information in real time.

A brief summary of the key features:

- Transformation of vast amounts of alarm data (or raw events) received from HPE TeMIP into meaningful information
- Use of the HPE Vertica database, optimized for data warehousing, data analytics and data reporting
- Optional activation of default summarizations batch jobs in order to populate new tables containing aggregated data about alarms or raw events
- Optional default FAS reports (built with HPE Unified OSS Console) based on those summarized tables in order to perform analysis and manage efficiently the telecommunication network

## 1.2 Architecture

A Fault Analytics and Statistics solution is composed of the following products:

- **HPE TeMIP** with TeMIP Alarm Handling analytics patch and corrective filter library
- **HPE UMB** Server containing Kafka/Zookeeper services
- **HPE OSS Analytics foundation**
- **HPE OSS Fault Analytics and Statistics** containing alarms/ raw events export application, default summarizations and off-the-shelf FAS UOCV2 reports
- **HPE Vertica**
- **HPE Unified OSS Console** (UOCv2)

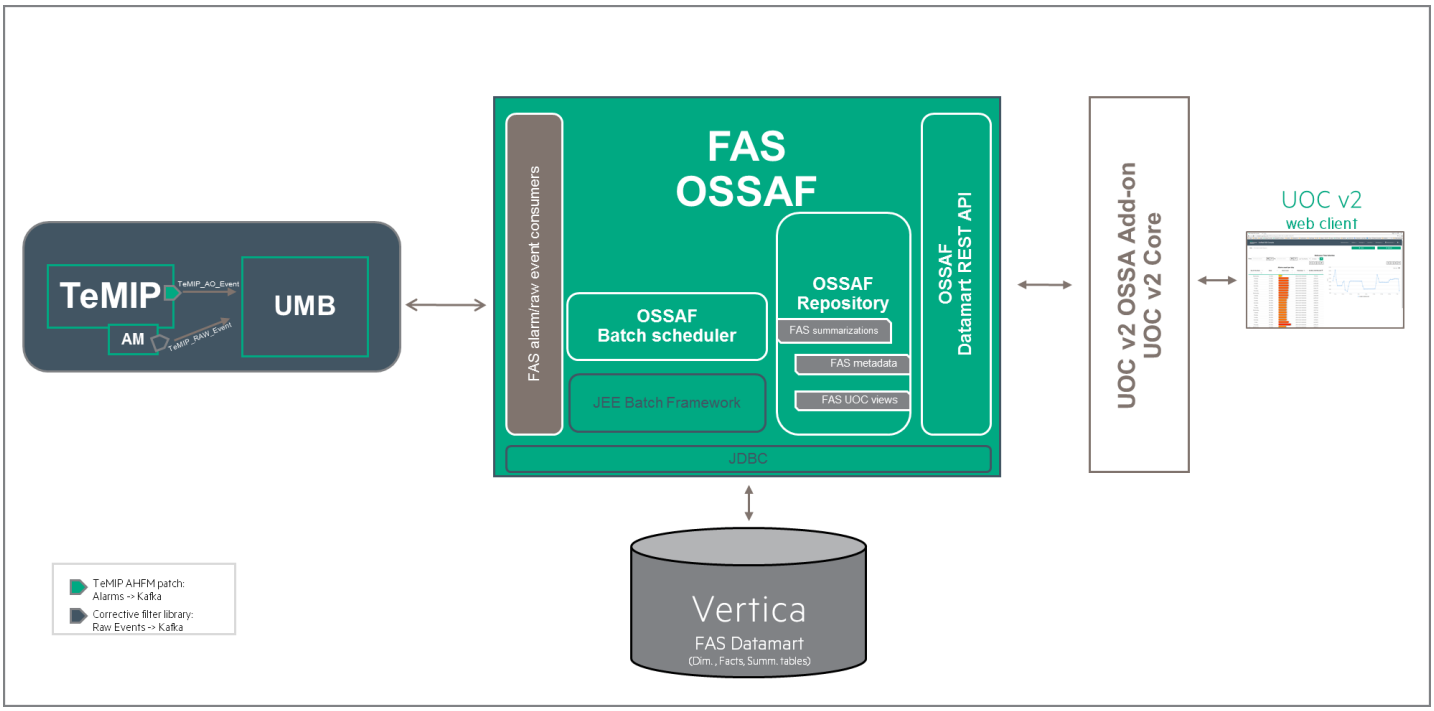


**Figure 1: OSS Fault Analytics & Statistics architecture**

### TeMIP patches

TeMIP AHFM patches must be installed on top of the TeMIP Framework Server 6.2 on RHEL platform in order to publish alarms in near real-time to the UMB Kafka broker.
In addition, a TeMIP corrective filter library must be installed in order to have TeMIP Access Modules publishing their raw events to UMB Kafka broker.

Those alarms / raw events exported into UMB Kafka will then be consumed by FAS.

### FAS

FAS relies on **HPE OSS Analytics Foundation (OSSAF)** which is a software component prerequisite for FAS.

FAS contains the export application which consumes and transforms alarms / raw events data received from TeMIP and populates the FAS datawarehouse based on HPE Vertica.

FAS also contains default summarizations: this is a description of a batch job that can be deployed and scheduled into OSSAF for aggregating data related to alarms / raw events.

The FAS metadata allows to describe the FAS datamart.
Once the FAS metadata are loaded, OSSAF Datamart REST API allows to access and perform analysis on FAS alarms/raw events stored in the Vertica database.
Once the FAS views and workspaces are loaded, **HPE Unified OSS Console v2** users can play with the FAS standard default analytical reports.

For more information about OSSAF, please refer to the *HPE OSS Analytics Foundation documentation*.
For more information about UOCv2, please refer to the *HPE UOC V2 User Guide*.

# Chapter 2
# Product installation

## 2.1 Code signing

The code signing procedure allows you to assess the integrity of the delivered product before installing it, by verifying the signature of the software packages.
HPE recommends using signature verification on its products.

### 2.1.1 Signature verification

#### 2.1.1.1 Import HPE public key

Perform the following steps to import the public key that is needed for verifying the delivered products. These steps requires to be logged as root.

1. Create a temporary directory where the public keys will be stored:

```
# mkdir -p signcheck
```

2. Download the public keys:

```
# wget -P signcheck/ https://ftp.hp.com/pub/keys/HPE-GPG-Public-Keys.tar.gz
```

3. Unzip and untar the downloaded file *HPE-GPG-Public-Keys.tar.gz*

4. Import the public key for rpm

```
# rpm --import signcheck/2BAF2262.pub
```

#### 2.1.1.2 FAS kit signature verification

To verify the integrity of the delivered product, perform the following steps:

```
$ rpm -Kv ossa-fault-1.2.0-MR.noarch.rpm
```

Check the command output. If signature verification completed successfully, the command output will be:

```
ossa-fault-1.2.0-MR.noarch.rpm:
    Header V3 RSA/SHA256 Signature, key ID 9bf2b0ca: NOKEY
    Header SHA1 digest: OK (6d27ab4d43c65fb05211f03798c08901e7f1b437)
    V3 RSA/SHA256 Signature, key ID 9bf2b0ca: NOKEY
    MD5 digest: OK (8166e5a61e0250bd1afc5e4030247ff2)
```

**NOTE:** For more information about signature verification procedure, please visit:

https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=HPLinuxCodeSigning2

# 2.2 Prerequisites

## 2.2.1 Hardware and operating systems prerequisites

The HPE Fault Analytics and Statistics is supported on Red Hat Enterprise Linux versions 6.8 (x86-64).

Before installing HPE Fault Analytics and Statistics, verify that your system meets the following minimum requirements.

**Table 2: Minimum hardware requirements for HPE Fault Analytics and Statistics on Linux**

| Hardware | Minimum requirement |
|---|---|
| CPU | 1 CPU 2.5 GHz 4 cores |
| RAM | 8 GB |
| Hard disk size | 50 GB |
| Network | 100 MB Ethernet |

For a real sizing exercise, please contact the FAS product manager.

## 2.2.2 Software prerequisites

### 2.2.2.1 TeMIP Framework Server 6.2 on RHEL

TeMIP Server 6.2 for RHEL platform must be setup in accordance to the information provided in TeMIP documentation (*TeMIP Framework Release Notes*, *TeMIP Framework Installation Guide*, etc.)
The TeMIP director must be configured with all the latest official patches.

In addition, the following TeMIP patches (or the superseding ones) must be installed and configured in order to export alarms in near real-time to the UMB Kafka broker:

| | |
|---|---|
| *TEMIPTFRLIN_00210* | TeMIP TFR TFC patch |
| *TEMIPTFRLIN-00234* | TeMIP TFR libraries patch |
| *TEMIPTFRLIN-00237* | AHFM FAS feature |

In addition, the following TeMIP patches (or the superseding ones) must be installed and configured in order to export raw events in near real-time to the UMB Kafka broker:

| | |
|---|---|
| *TEMIPTFRLIN_00240* | Corrective Filter library patch for publishing events to Kafka |

**NOTE:** For more details, please refer to individual patch readme files.

### 2.2.2.2 UMB Server

UMB Server containing Kafka/Zookeeper broker are prerequisites for FAS:

- `umb-kafka-package-1.1-linux.tar`
- `umb-zookeeper-package-1.1-linux.tar`

### 2.2.2.3 Java

Please refer to the *HPE OSS Analytics Foundation Installation Guide*.

### 2.2.2.4 OSS Analytics Foundation

OSS Analytics Foundation is a prerequisite for FAS.

**CAUTION:** FAS 1.2 absolutely needs to be installed on top of **OSS Analytics Foundation 1.1.4**.

So, please refer to the corresponding *HPE OSS Analytics Foundation Installation Guide*.

### 2.2.2.5 Vertica database

FAS stores domain specific data into a Vertica datawarehouse.
Thus, it requires the HPE Vertica 7.2.x database server and client to be installed.

Please refer to HPE OSS Analytics Foundation documentation for more information.

### 2.2.2.6 Unified OSS Console (UOC v2)

In order to visualize the default FAS analytical reports, you need to install the Unified OSS Console v2 product and the UOC OSSA Add-on.
Please refer to Unified OSS Console Installation Guide for information about UOC installation.
Please refer to OSS Analytics foundation Installation Guide for information about UOC OSSA Add-on installation.

## 2.3 Example of FAS installation in production environment

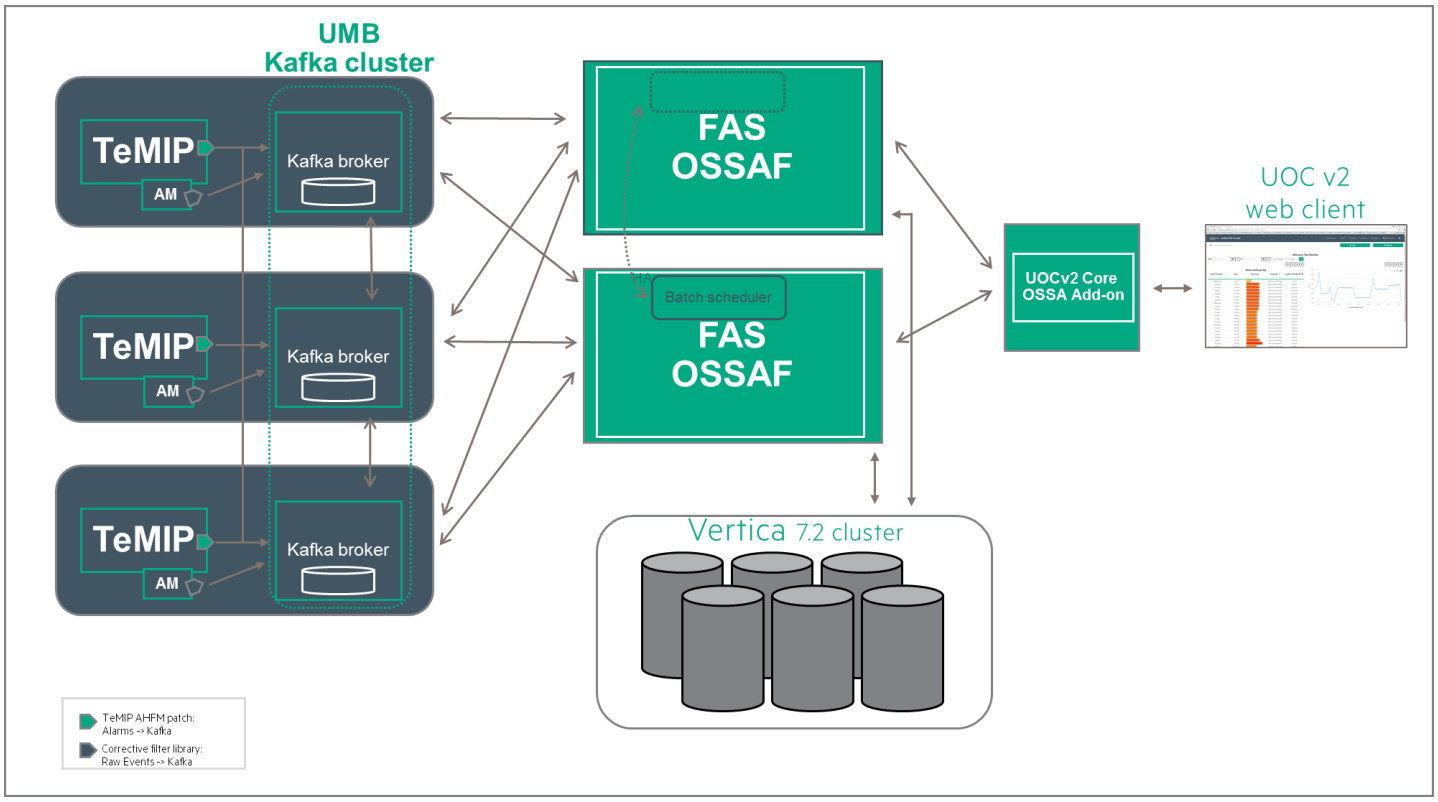


**Figure 2: Example of FAS installation in production**

With this kind of deployment one can benefit from **high-availability** of the components in an **active-active** mode:

- All Kafka brokers of the **UMB Kafka cluster** are handling TeMIP alarms and raw events. And, in case of broker failure, the consumption continues and no data is lost thanks to the replication factor set on Kafka.
- All **FAS servers** are consuming alarms and raw events as soon as all the FAS servers are in the same Kafka consumer group. If a FAS server fails, the whole consumption is then processed by the remaining one.
  Note also, that the **OSSAF batch scheduler** (triggering the FAS summarization batch job processing) will automatically be executed on the remaining FAS server.
- Vertica cluster composed of several Vertica nodes also works in active-active mode.

# 2.4 Installation of prerequisites products

## 2.4.1 TeMIP patches installation

The patches mentioned in the section TeMIP patches concerning **alarms publications** must be installed on all TeMIP servers running Alarm Handling.

The patches mentioned in the section TeMIP patches concerning **raw events publications** must be installed on all servers running TeMIP Access Modules which must publish their events to UMB.

**NOTE:** Please refer to the corresponding patch texts for having all the details of **TeMIP patches installation**.

## 2.4.2 UMB server installation

Please refer to the *HPE Unified Mediation Bus - Installation and Configuration Guide*, for the installation of UMB-Kafka / UMB-Zookeeper broker components.

WARNING:
In addition to the configuration of Kafka / Zookeeper done while installing the UMB packages, there are some additional configurations to be done in */var/opt/UMB/kafka/config/server.properties*

- The topics handled by Kafka and FAS must be controlled so you must modify the parameter `auto.create.topics.enable` to `auto.create.topics.enable=false`

- Concerning the retention of data by Kafka. Both `log.retention.hours` and `log.retention.bytes` can be set in order to control the retention of data by Kafka. But note that when both are set, data are deleted when either limit is exceeded. So, you can set a `log.retention.hours` to 1 month for example: `log.retention.hours=720` and you can accurately set the `log.retention.bytes` parameter depending on your disk capacity.

Warning: `log.retention.bytes` parameter is the amount of data to retain in the log for each topic-partitions. So, multiply this number by 64 (default number of partition within FAS) for knowing the disk space that will be used by the Kafka retained data. This will allow you to determine which value to set for `log.retention.bytes` parameter depending on your disk capacity.
(note that `log.retention.bytes` parameter * 64 / *alarm size* / *alarm rate* should give you an approximation of the history time window size of the alarms retained)

## 2.4.3 OSS Analytics Foundation installation

Please follow the instructions given in the *HPE OSS Analytics Foundation Installation and Configuration Guide*.

# 2.5 FAS installation

## 2.5.1 Unix user

The *'ossa'* unix user should already be created after having installed the OSS Analytics Foundation.

## 2.5.2 FAS kit installation

**CAUTION:** Remember that FAS 1.2 absolutely needs to be installed on top of OSS Analytics Foundation 1.1.4.

Please install the FAS rpm package on your linux system. As *root* user:

```
# rpm -ivh ossa-fault-1.2.0-MR.noarch.rpm
```

**NOTE:** The default installation location of the RPM package is */opt/ossa/*
Use the standard rpm `--prefix` options for installing FAS in another directory of your choice.

## 2.5.3 FAS environment variables settings

Now, all the remaining steps must be executed with '*ossa*' unix user.

So, as *ossa* user, edit the installation configuration file:

*/opt/ossa/ossa_fault.cfg*

and modify the environment variables defined according to your configuration.

**NOTE:** In case you are installing FAS for the first time, you can create this ***ossa_fault.cfg*** configuration file by taking a default example delivered. You can:

```
cp /opt/ossa/ossa_fault.cfg.dist /opt/ossa/ossa_fault.cfg
```

and then modify the ***ossa_fault.cfg***

**Table 3: *ossa_fault.cfg* parameters description**

| Parameter | Expected value |
|---|---|
| *JBOSS_SERVER_OPTS* | In case you want to add some FAS JBoss Wildfly server options you can specify them with this parameter.<br>For example, if you want to limit the Kafka consumer groups processed by FAS, you can specify them explicitly with:<br>`' -DsupportedConsumerGroupIds=["fas-ao-01","fas-re-01"]'` |
| *OSSA_DB_TYPE_02* | The data warehouse database type.<br>In this version of FAS only "vertica" is supported. |
| *OSSA_DB_HOST_02* | The hostname or IP address of the database server hosting the data warehouse. |
| *OSSA_DB_NAME_02* | The name of the database hosting the data warehouse. |
| *OSSA_DB_PORT_02* | The port of the database. |
| *OSSA_DB_USER_02* | The name of the database user to access the data warehouse. |
| *OSSA_DB_PASSWORD_02* | The password of the database user to access the data warehouse. |
| *OSSA_DB_URL_02* | In case database connection needs some extra parameters, you can fully specify it with this parameter.<br>Example, for setting load balancing, you can specify:<br>`"jdbc:${OSSA_DB_TYPE_02}://${OSSA_DB_HOST_02}:${OSSA_DB_PORT_02}/${OSSA_DB_NAME_02}?BackupServerNode=mynode1.mydomain.com,mynode2.mydomain.com&ConnectionLoadBalance=1"` |
| *OSSA_DS_NAME_02* | The name of the FAS JEE data source to use in the JEE container.<br>Keep the default values for standard usage. |
| *OSSA_DS_MAX_POOL_SIZE_02* | The maximum number of connections created in the FAS JEE connection pool for accessing the data warehouse: theorically, the number must correspond at least to the number of Kafka consumer threads (for alarms consumption and for raw events consumption). |

For example:

```
OSSA_DB_TYPE_02=vertica
OSSA_DB_HOST_02=myDBHost
OSSA_DB_NAME_02=OSSA
OSSA_DB_PORT_02=5433
OSSA_DB_USER_02=FAS
OSSA_DB_PASSWORD_02=FASpwd
OSSA_DS_NAME_02=OssaFaultDS
OSSA_DS_MAX_POOL_SIZE_02=32
```

Then, source the OSSAF environment in order to take into account your settings:

```
$ source /opt/ossa/bin/ossa_env.sh
```

## 2.5.4 FAS database user

The first database operation to be performed is to create the FAS database user.

**NOTICE:** While installing OSSA Foundation, the default and recommended database name used was: "OSSA". In case you have not chosen this database name, please modify the ${OSSA_HOME}/ddl/create_user_fas.sql by replacing in `GRANT CREATE ON DATABASE OSSA TO :fas_user;` the `OSSA` by your database name.

The creation of the FAS database user must be performed with the following command:

```
$ vsql -d ${OSSA_DB_NAME_02} -h ${OSSA_DB_HOST_02} -U dbadmin  -w
<dbadminpwd> -v ossa_user=${OSSA_DB_USER_01} -v fas_user=${OSSA_DB_USER_02} -
f ${OSSA_HOME}/ddl/create_user_fas.sql
```

Note that a default dedicated Vertica resource pool *pool_fas* will also be created.
Please refer to next section "*Vertica resource pools configuration*", for details information about its configuration.

## 2.5.5 FAS database schema

Please execute the following commands for creating the FAS datamart.

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U dbadmin -w <dbadminpwd>
-v fas_user=${OSSA_DB_USER_02} -v ossa_user=${OSSA_DB_USER_01} -f
${OSSA_HOME}/ddl/create_schema_fault.sql
```

The following command creates the datamart tables related to Alarms:

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -v fas_user=${OSSA_DB_USER_02} -f
${OSSA_HOME}/ddl/fault_datamart.sql
```

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -v fas_user=${OSSA_DB_USER_02} -f
${OSSA_HOME}/ddl/fault_datamart_summ.sql
```

The following command creates the datamart tables related to Raw Events:

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -v fas_user=${OSSA_DB_USER_02} -f
${OSSA_HOME}/ddl/raw_event_datamart.sql
```

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -v fas_user=${OSSA_DB_USER_02} -f
${OSSA_HOME}/ddl/raw_event_datamart_summ.sql
```

This last step is required to allow the OSSA database user to access tables created in FAS schema. (FAS reports rely on OSSAF Datamart REST API, which performs SELECT queries using database user OSSA)

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U dbadmin -w <dbadminpwd>
-v ossa_user=${OSSA_DB_USER_01} -v fas_user=${OSSA_DB_USER_02} -f
${OSSA_HOME}/ddl/grant_to_ossa.sql
```

**NOTE:** In case you want to process user defined attributes, you must prepare additional and specific SQL script. Please refer to the *HPE OSS Fault Analytics and Statistics Customization guide* for more information.

## 2.5.6 FAS configuration

For FAS configuration, please perform the following commands:

```
$ ossa_config.sh
```

then,

```
$ ossa_config_fault.sh
```

**CAUTION:** In case you have modified *ossa.cfg* for OSSAF configuration or *ossa_fault.cfg* for FAS configuration, please, stop OSSAF, and then execute: *${OSSA_HOME}/bin/ossa_config.sh*

then, you can start the application server FAS/OSSAF:

```
$ jbossstart
```

## 2.5.7 FAS alarm/raw event export configuration

Here are the main properties that can be defined for FAS alarm / raw event export application:

**Table 4: FAS alarm/raw event export configuration parameters**

| Property | Sub property | Example of value | Description |
|---|---|---|---|
| consumerGroupConfigs / consumerGroupProperties | | | |

| | | | |
|---|---|---|---|
| | **group.id** | *myproject-cg-kafka_fas* | A string that uniquely identifies the Kafka consumer group to which this consumer belongs.<br><br>Recommended naming convention: *<project>-cg-<from-kafka-cluster-name>-<to-consumer-name>*<br>This is useful for troubleshooting in case several FAS applications consume alarm messages from several kafka clusters. |
| | **zookeeper.connect** | *myzkhost:2181* | Specifies the ZooKeeper connection string in the form *hostname:port* where *host* and *port* are the host and port of a Zookeeper server.<br><br>If there are several ZooKeeper nodes, use the form *hostname1:port1,hostname2:port2...* |
| | **auto.commit.enable** | *false* | This property is enforced to false by FAS alarm export application whatever the value specified in config, it will not be taken into account. This is because the offset must be committed by FAS alarm export application explicitly and never auto committed by Kafka. |
| | **auto.offset.reset** | *smallest* | Determine what to do when there is no initial offset in Zookeeper or if an offset is out of range:<br>- *smallest* : automatically reset the offset to the smallest offset<br>- largest : automatically reset the offset to the largest offset<br>- anything else: throw exception to the consumer.<br>The default value is *smallest*. |
| consumerGroupConfigs / **consumerThreadNumber** | | *4* | The number of consumer threads for the FAS Alarm Export. |
| consumerGroupConfigs / **writeBatchSize** | | *10000* | As soon as consumer thread has accumulated *writeBatchSize* alarms, then the batch is flushed to DB.<br>Default value is 1000. |
| consumerGroupConfigs / **writeIntervalMaxSeconds** | | *60* | Data is flushed to DB after this delay, even if less that *writeBatchSize* alarms have been accumulated.<br>Default value is 60. |
| consumerGroupConfigs / **destDataSourceName** | | *OssaFaultDS* | Name of the datasource where the data is pushed.<br>By default, it corresponds to the standard datasource name as defined in a standard installation: *OssaFaultDS* |

| | | | |
|---|---|---|---|
| consumerGroupConfigs / **destSchema** | | *OSSA_FAULT* | Name of the database schema where the data is pushed.<br>By default, it corresponds to the standard schema name as defined in a standard installation: *OSSA_FAULT* |
| consumerGroupConfigs / **tmMaxAcceptedDurationMinutes** | | *5* | This parameter is related to the Vertica activity check done during FAS Alarm Export in order to avoid 'burst' of data written onto Vertica.<br>This consists in checking if Vertica Tuple Mover operation is running on the destination table, and since how much time.<br>This parameter defines the maximum number of minutes that we accept the Vertica Tuple Mover to run. (default value is 5 minutes)<br>If Vertica Tuple Mover is running since more than *verticaTmMaxAcceptedDurMins*, the processing is suspended until Vertica Tuple Mover operations will not be so intensive. |
| **beforeMergeToFfSql** | | *DELETE FROM TMP_FF WHERE ORIGINALEVENTTIME < getdate() - 180* | This parameter allows to filter out alarms from the FAS Alarm Export.<br>The value must be a valid SQL statement working on *TMP_FF* table (which has the same structure than *FCT_FAULT* table).<br>By default, there is no filter defined. |
| **temipClassSpecificProblems** | | | Some access modules and associated TeMIP classes use specific problems in degenerated way to encode something else than specific problems, e.g. notification id.<br>In such case the mapping of specific problems should be avoided.<br>Those following properties allow to decide whether the specific problems are exported to Vertica or not. |
| | **temipClass** | *OSI_SYSTEM.TESTOBJ* | Name of a TeMIP class on which you want to configure specific problem mapping. |
| | **supportsSP** | *true* | Decide whether the Specific Problems attribute related to the *temipClass* above should be persisted to Vertica table *DIM_SPECIFICPROBLEMS* or not.<br>Default value is false. |

In case you want to configure other Kafka consumer parameters, please refer to the following documentation:
https://kafka.apache.org/08/configuration.html

How to set those parameters? Here is an example where FAS consumes and exports Alarms and Raw Events:

```
$ ${OSSA_HOME}/bin/ossa-repo.sh setParam FAS AlarmExportConfig '
{ "consumerGroupConfigs" : [
    { "topicId" : "TeMIP_AO_Event",
      "destDataSourceName" : "OssaFaultDS",
      "destSchema" : "FAS",
      "consumerThreadNumber" : 4,
      "writeBatchSize" : "10000",
      "consumerGroupProperties" : { "group.id" : "fas-ao-01",
                                    "zookeeper.connect" : "host1:2181,host2:2181,host3:2181",
                                    "auto.commit.enable" : "false",
                                    "auto.offset.reset" : "smallest"} },
    { "topicId" : "TeMIP_RAW_Event",
      "destDataSourceName" : "OssaFaultDS",
      "destSchema" : "FAS",
      "consumerThreadNumber" : 8,
      "writeBatchSize" : "10000",
      "consumerGroupProperties" : { "group.id" : "fas-re-01",
                                    "zookeeper.connect" : "host1:2181,host2:2181,host3:2181",
                                    "auto.commit.enable" : "false",
                                    "auto.offset.reset" : "smallest" } } ] } '
```

Then, reload the modification within OSSAF repository:

```
ossa-repo.sh reload
```

At this step, you have configured FAS in order to consume alarms / raw events from UMB and publish them to Vertica.

The next sections provide information about configuration of FAS standard reports and their required back end processing: the FAS batch jobs.

## 2.5.8 Load FAS batch jobs

Please follow the steps below in order to load the standard FAS batch jobs.

**NOTE:** In case you want to process user defined attributes in summarizations, you must prepare additional and specific SQL scripts and xml files. Please refer to the *HPE OSS Fault Analytics & Statistics Customization guide* for more information.

In the terminal where you sourced *ossa_env.sh*, as *ossa* user, please perform:

```
$ cd ${OSSA_HOME}/bin
$ ./fas_load_batchJobs.sh
```

## 2.5.9 Load default FAS UOCv2 standard reports

In the terminal where you sourced *ossa_env.sh*, as *ossa* user, load the default FAS UOCv2 reports (views and workspaces) in OSSAF repository.

```
$ cd ${OSSA_HOME}/repo-fas/
```

```
$ ${OSSA_HOME}/bin/ossa-repo.sh loadMetadataViewsWks ${OSSA_HOME}/repo-
fas/metadata/ossa_fault_metadata.xml ${OSSA_HOME}/repo-fas/ui/views.json
${OSSA_HOME}/repo-fas/ui/workspaces.json
$ ossa-repo.sh reload
```

Once this command is run, you will be able to view the reports in UOCv2.

## 2.5.10 Verify your installation

You can run the following command in order to check that the **rpm packages** are installed:

```
$ rpm -qa | grep ossa
```

The output must display the ossa-server and the ossa-fault packages installed:

```
ossa-server-1.1.4-MP.noarch
ossa-fault-1.2.0-MR.noarch
```

Then, you can run the following command in order to check that the application server is running and that the *ossa-server* and the *ossa-fault* **applications are deployed**:

```
$ ${JBOSS_HOME}/bin/jboss-cli.sh --connect --controller=${JBOSS_HOST}:`expr
9990 + ${JBOSS_PORT_OFFSET}` -c 'ls deployment'
```

The output should display the ossa-restapi and ossa-fault ear:

```
ossa-fault-1.2.0.ear
ossa-server.ear
```

Then, you can check that the FAS standard batch jobs are loaded into OSSAF repository and scheduled regularly: for this, you can open the OSSAF Administration console (please refer to *HPE OSS Analytics Foundation Installation Configuration and Administration Guide*)

And finally, you can check that the **FAS reports** are available in UOCv2 workspaces.

## 2.5.11 TCP port used

Here are the default TCP ports used within a FAS solution:

- Zookeeper client port: 2181
  (+ additional ports `2888`, `3888` in case of high availability configuration)
- Kafka client port: 9092
- OSSA REST Server client port: 8080
- Vertica client port: 5433
- Unified OSS Console client port: 3000

For configuring those ports please refer to:

- UMB Installation & Configuration Guide for Kafka/Zookeeper
- OSSA Foundation Installation & Configuration Guide for OSSAF REST Server
- Vertica documentation for Vertica
- Unified OSS Console Installation & Configuration Guide for UOC

# Chapter 3
# Product uninstallation

For uninstalling FAS product, please follow those steps. In the terminal where you sourced *ossa_env.sh*, as ***ossa*** user:

```
jbossstop
```

Then, switch to ***root*** user and run the following command:

```
rpm -ev ossa-fault-1.2.0-MR.noarch
```

**IMPORTANT:** The uninstallation does not remove the ***ossa_fault.cfg*** file, which contains your FAS configuration. If you plan to install a new version of FAS, keep this file at its current location. During the next installation, the file will not be overwritten and will be used as it is now. In that way, the same configuration is used for your next installation.

For uninstalling the other products of the solution, please refer to the related documentation.

# Chapter 4
# Product administration

## 4.1 Configure TeMIP publication to Kafka

### 4.1.1 Configure TeMIP alarms publication

Here are the operations to perform in order to enable TeMIP alarms from operation contexts to be published to Kafka:

- Create a Kafka topic concerning TeMIP alarms:

```
<UMB_installation_dir>/bin/kafka-topics.sh --create --zookeeper <Zookeeper
host>:2181 --replication-factor 3 --partitions 64 --topic TeMIP_AO_Event
```

  - Replication-factor is the number of replica that you want and must be less than or equal to the number of Kafka brokers
  - Recommended number of partitions in Kafka is 64
  - the topic name must contain the following string: *TeMIP_AO_Event*

- Then, concerning TeMIP AHFM configuration please refer to the patch readme file (patch referenced in this document at: *Software prerequisites / TeMIP Framework Server 6.2*)

### 4.1.2 Configure TeMIP AM Raw Events publication

Here are the operations to perform in order to enable raw events from TeMIP AM to be published to Kafka:

- Create a Kafka topic concerning TeMIP AM raw events:

```
<UMB_installation_dir>/bin/kafka-topics.sh --create --zookeeper <Zookeeper
host>:2181 --replication-factor 3 --partitions 64 --topic TeMIP_RAW_Event
```

  - Replication-factor is the number of replica that you want and must be less than or equal to the number of Kafka brokers
  - Recommended number of partitions in Kafka is 64
  - the topic name must contain the following string: *TeMIP_RAW_Event*

- Then, concerning TeMIP AM configuration please refer to the corrective filter patch readme file (patch referenced in this document at: *Software prerequisites / TeMIP Framework Server 6.2*)

## 4.2 Administrate OSS Analytics Server

Please follow the HPE OSS Analytics Foundation Installation and Administration guide.

# 4.3 Administrate FAS

## 4.3.1 Stop FAS

In the terminal where you sourced *ossa_env.sh*, as *ossa* user, execute:

```
jbossstop
```

This will stop the whole OSS Analytics Server, including FAS.

## 4.3.2 Start FAS

In the terminal where you sourced *ossa_env.sh*, as *ossa* user, execute:

```
jbossstart
```

This will start the whole OSS Analytics Server, including FAS.

## 4.3.3 Check server process

In the terminal where you sourced *ossa_env.sh*, as *ossa* user, execute:

```
$ jbossshow
```

If OSSA/FAS server is running, the output will display process information: pid, command.
If OSSA/FAS server is not running, the output will be empty.

## 4.3.4 Manage data retention in FAS Datamart

You can manage the retention of your data in FAS datamart thanks to the cleanup batch jobs located in the directory:
`${OSSA_HOME}/repo-fas/batch/`

Concerning alarms:

| CLEANUP on tables | job cleanup definition file | job cleanup scheduling file |
|---|---|---|
| `FCT_FAULT` | FAScleanupFaultJob.xml | BATCH_FAScleanupFault.json |
| `SUMM_HOURLY_...` | FAScleanupFaultSummHourlyJob.xml | BATCH_FAScleanupFaultSummHourly.json |
| `SUMM_DAILY_...` | FAScleanupFaultSummDailyJob.xml | BATCH_FAScleanupFaultSummDaily.json |
| `SUMM_WEEKLY_...` | FAScleanupFaultSummWeeklyJob.xml | BATCH_FAScleanupFaultSummWeekly.json |
| `SUMM_MONTHLY_...` | FAScleanupFaultSummMonthlyJob.xml | BATCH_FAScleanupFaultSummMonthly.json |

Concerning raw events:

| CLEANUP on tables | job cleanup definition file | job cleanup scheduling file |
|---|---|---|
| `FCT_RAW_EVENT` | FAScleanupRawEventJob.xml | BATCH_FAScleanupRawEvent.json |
| `SUMM_HOURLY_RAW_EVENT...` | FAScleanupRawEventSummHourlyJob.xml | BATCH_FAScleanupRawEventSummHourly.json |

| | | |
|---|---|---|
| SUMM_DAILY_RAW_EVENT... | FAScleanupRawEventSummDailyJob.xml | BATCH_FAScleanupRawEventSummDaily.json |
| SUMM_WEEKLY_RAW_EVENT... | FAScleanupRawEventSummWeeklyJob.xml | BATCH_FAScleanupRawEventSummWeekly.json |
| SUMM_MONTHLY_RAW_EVENT... | FAScleanupRawEventSummMonthlyJob.xml | BATCH_FAScleanupRawEventSummMonthly.json |

In those files, there are default values for data retention periods and default schedules defined.
But, you can define your own configuration.

1. Define your data retentions
   First of all, modify the value X of the property `XXXXRetentionYYYY` defined in all the `FAScleanupFault<tables>Job.xml` files in order to match your requested retention periods.
   The property `<property name="XXXXRetentionYYYY" value="X"/>` defines the retention period for the summarization.

   XXXX is the kind of summarization :
   fault for FCT_FAULT table
   summHourly for SUMM_HOURLY_... tables
   summDaily for SUMM_HOURLY_... tables
   summWeekly for SUMM_HOURLY_... tables
   summMonthly for SUMM_HOURLY_... tables

   YYYY is the unit of the value: Month or Day or Year

   X is the value to modify

2. Define your execution schedules
   Then, you can define the schedule of the execution of the batch jobs by modifying the section "`batchSchedule`" within the files `BATCH_FAScleanup<tables>Summ<granularity>.json`

3. Schedule your cleanup batch jobs

   You can now load your batch job definition, and your batch job schedules.

   Concerning alarms:

```
ossa-repo.sh loadParam FAS FAScleanupFault.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupFaultJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupFault.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupFault.json


ossa-repo.sh loadParam FAS FAScleanupFaultSummHourlyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupFaultSummHourlyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupFaultSummHourly.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupFaultSummHourly.json


ossa-repo.sh loadParam FAS FAScleanupFaultSummDailyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupFaultSummDailyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupFaultSummDaily.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupFaultSummDaily.json


ossa-repo.sh loadParam FAS FAScleanupFaultSummWeeklyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupFaultSummWeeklyJob.xml
```

```
ossa-repo.sh loadParam FAS BATCH_FAScleanupFaultSummWeekly.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupFaultSummWeekly.json


ossa-repo.sh loadParam FAS FAScleanupFaultSummMonthlyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupFaultSummMonthlyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupFaultSummMonthly.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupFaultSummMonthly.json

ossa-repo.sh reload
```

Concerning raw events:

```
ossa-repo.sh loadParam FAS FAScleanupRawEvent.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupRawEventJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupRawEvent.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupRawEvent.json


ossa-repo.sh loadParam FAS FAScleanupRawEventSummHourlyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupRawEventSummHourlyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupRawEventSummHourly.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupRawEventSummHourly.json


ossa-repo.sh loadParam FAS FAScleanupRawEventSummDailyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupRawEventSummDailyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupRawEventSummDaily.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupRawEventSummDaily.json


ossa-repo.sh loadParam FAS FAScleanupRawEventSummWeeklyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupRawEventSummWeeklyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupRawEventSummWeekly.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupRawEventSummWeekly.json


ossa-repo.sh loadParam FAS FAScleanupRawEventSummMonthlyJob.xml \
          ${OSSA_HOME}/repo-fas/batch/FAScleanupRawEventSummMonthlyJob.xml

ossa-repo.sh loadParam FAS BATCH_FAScleanupRawEventSummMonthly.json \
          ${OSSA_HOME}/repo-fas/batch/BATCH_FAScleanupRawEventSummMonthly.json

ossa-repo.sh reload
```

Your cleanup batch jobs will then be executed regularly.

# Chapter 5
# Product tuning

Depending on the FAS solution you have deployed and configured, there are some parameters that must be tuned in order to optimize the application execution.

## 5.1 JBoss

### 5.1.1 JBoss OSS Analytics datasource maximum pool size

In the configuration file *${OSSA_HOME}/ossa.cfg*, the value of the configuration parameter *OSSA_DS_MAX_POOL_SIZE_01* must correspond to the targeted number of concurrent users from UOC that will access the OSS Analytics server.
Note that this number must also correspond to the Vertica **OSSA** resource pool parameter ***plannedconcurrency*** (see sections below).

### 5.1.2 JBoss FAS datasource maximum pool size

In the configuration file *${OSSA_HOME}/ossa_fault.cfg*, the value of the configuration parameter *OSSA_DS_MAX_POOL_SIZE_02* must correspond to the sum of:
- number of FAS consumer threads (see *consumerThreadNumber* parameter setting in the paragraph *Configuring FAS parameters*)
- parallel connections for FAS summarizations ( 3 for the default 3 categories of alarm data summizations + 1 for the default raw event summarization).

So, if *consumerThreadNumber* = 4 and all default summarizations are activated (4 threads used), *OSSA_DS_MAX_POOL_SIZE_02* must be equals to 8.
Note that this number must also correspond to the Vertica **FAS** resource pool parameter **plannedconcurrency** (see sections below)

**CAUTION:** when modifying *ossa.cfg* or *ossa_fault.cfg*, OSSAF must be re-configured, so please, stop OSSAF, and execute: *${OSSA_HOME}/bin/ossa_config.sh*

## 5.2 Vertica

### 5.2.1 Vertica resource pools configuration

You must absolutely define:
- 1 resource pool for OSSA Vertica user: this resource pool must be used only by OSSA user
- 1 resource pool for FAS Vertica user: this resource pool must be used only by FAS user

If you used the default OSSA/FAS database creation scripts, you have default *pool_ossa* and *pool_fas* Vertica resource pools defined.

Here are the recommendations concerning Vertica resource pools:
- The Vertica tuple mover resource pool named *"tm"* must be the only one with *HIGH runtimepriority*.
- Thus, your user defined resource pool (*pool_fas*) must have runtime priority set to *MEDIUM*, with *runtimeprioritythreshold* = 0, so that queries are automatically assigned *MEDIUM* priority.

This means that you can set for example:
*pool_FAS* resource pool *runtimepriority* to *MEDIUM*
*pool_OSSA* resource pool *runtimepriority* to *LOW*

- *pool_OSSA* resource pool must have *execution parallelism = 1*, so that you limit the number of threads for each single query, this prevents contention in Vertica, and thus it allows a maximum number of parallel requests to be executed.

- Maximize the usage of the available RAM by defining accordingly *memorysize* (which is the memory size allocated specifically to a resource pool) in order to have all available RAM being potentially used across resource pools: for user defined resource pools (*pool_OSSA*/*pool_FAS*) but also Vertica system resource pool.
Also, define accordingly the *maxmemorysize*, which allows to limit how much memory the resource pool can borrow from the GENERAL pool.

- define correctly the *priority* (which prioritize the use of GENERAL pool resources), by giving high priority to the tuple mover. For example:

- sysquery priority = 110
- recovery priority = 107
- tm priority = 105
- fas priority = 50
- ossa priority = 10
- other priority = 0

- The parameters *plannedconcurrency* / *maxconcurrency* must be set in order to limit the number of concurrent requests within each pool (refer to settings JBoss OSSAF / FAS datasource maximum pool size defined above).

Concerning the tuple mover resource pool you can set for example:
tuple mover *plannedconcurrency* / *maxconcurrency* 25 / 25
Do not set too high values in order to avoid any I/O bottle neck.

Here are some examples of SQL requests for setting those parameters discussed above:

```
ALTER RESOURCE POOL POOL_OSSA MEMORYSIZE '40G' MAXMEMORYSIZE '60G' EXECUTIONPARALLELISM 1
MAXCONCURRENCY 40 PLANNEDCONCURRENCY 32 RUNTIMEPRIORITY LOW RUNTIMEPRIORITYTHRESHOLD 0
PRIORITY 0;

ALTER RESOURCE POOL POOL_FAS  MEMORYSIZE '40G' MAXMEMORYSIZE '60G' EXECUTIONPARALLELISM 2
MAXCONCURRENCY 40 PLANNEDCONCURRENCY 35 RUNTIMEPRIORITY MEDIUM PRIORITY 50;

ALTER RESOURCE POOL TM MEMORYSIZE '10G' EXECUTIONPARALLELISM 2 MAXCONCURRENCY 6
PLANNEDCONCURRENCY 6 RUNTIMEPRIORITY HIGH PRIORITY 100;
```

- The parameter *runtimecap* can be defined for the *pool_OSSA* resource pool.

It defines the maximum duration for queries before Vertica automatically cancel them.

This could prevent undesirable long running SQL requests coming from 'incorrect' user defined UOC dashboards, from which requests are scanning unintentionally whole content of huge tables of your datamart for example.

For more detailed information about resource pool settings you can refer to Vertica documentation:

http://my.vertica.com/docs/7.2.x/HTML/index.htm#Authoring/AdministratorsGuide/ResourceManager/GuidelinesForSettingPoolParameters.htm

**CAUTION:** Depending on your deployed solution in term of concurrent access to Vertica database (OSSA pool size used for accessing the FAS reports, FAS pool size for exporting alarms into Vertica and transforming data thanks to batch jobs, other external applications accessing Vertica database), you might also need to modify the Vertica configuration parameter: *MaxClientSessions*.

This can be checked with:

```
SELECT current_value FROM CONFIGURATION_PARAMETERS WHERE parameter_name='MaxClientSessions';
```

and modified like this (for example):

```
ALTER DATABASE OSSA SET MaxClientSessions = 500;
```

## 5.2.2 Vertica load balancing

The executions of requests within Vertica are done across all the nodes of the cluster, but by default, the requests come into Vertica onto one unique node.
As an option, you can configure Vertica with load balancing in order to spread the connections evenly over the nodes in the Vertica cluster.
This option must be set at both the server and client side. If not set for a client, the client will continue to connect to the originally defined node.

To set load balancing on the servers, you must execute on the database, as *dbadmin*:

```
SELECT SET_LOAD_BALANCE_POLICY('ROUNDROBIN');
```

On client side, this can be interesting to set load balancing for requests coming from OSSAF REST Server where there could be a high number of parallel requests. For doing this, please modify the OSSAF Server configuration file (*ossa.cfg*) and add the parameter *ConnectionLoadBalance=1* in the connection url.

Here is an example:

```
OSSA_DB_URL_01="jdbc:${OSSA_DB_TYPE_01}://${OSSA_DB_HOST_01}:5433/${OSSA_DB_N
AME_01}?BackupServerNode=<my2ndVerticaNode>,
<my3rdVerticaNode>&ConnectionLoadBalance=1"
```

**CAUTION:** In case you have modified *ossa.cfg* for OSSAF configuration, please, stop OSSAF, and execute: *${OSSA_HOME}/bin/ossa_config.sh*

# Chapter 6
# Troubleshooting

## 6.1 Debug TeMIP Analytics

TeMIP traces are stored in the directory */var/opt/temip/trace*

Enable tracing for all the processes in AHFM:
Enable the trace mask (see below) for getting the trace logs from Alarm Handling FM. This needs restart of AHFM. It produces the traces for AHFM foreground process and all the operation contexts after AHFM restart.

```
# manage set mcc 0 app alarm_handling_fm trace mask=0x60000000
# manage restart mcc 0 app alarm_handling_fm
```

Tracing a particular process in AHFM:
It is possible to set the trace mask for a running process. The trace mask is applicable only during the life time of the process. Find the process ID using */usr/opt/bin/temip_show* command, then:

```
# manage set mcc 0 app alarm_handling_fm process <PID> trace mask=0x60000000
```

Note that when the process is restarted the trace mask will be reset to previous value.

## 6.2 General FAS troubleshooting

### 6.2.1 Log file

You can find FAS log file at:

```
${JBOSS_HOME}/standalone/log/ossa_fault.log
```

There might be several rolling log files in the directory. When issue occurs, first identify the occurring time and then select the right log file.

### 6.2.2 Log level

In order to perform deeper investigations concerning possible FAS errors, you can set more detailed log level on FAS application.

First, source *${OSSA_HOME}/bin/ossa_env.sh*, then you can invoke shell functions for setting / getting log level.

For example:

```
$ ossa_fault_set_log DEBUG
$ ossa_fault_get_log
```

You can set specific log levels for performance (write) investigations:

```
$ ossa_fault_set_log_write_perf DEBUG
$ ossa_fault_get_log_write_perf
```

You can set specific log levels for investigations about alarm messages received in FAS:

```
$ ossa_fault_set_log_receive_alarm DEBUG
$ ossa_fault_get_log_receive_alarm
```

To enable logging of alarm message, you can set level name to *TRACE*.

**CAUTION:** On a production environment, take care of not setting too much detailed traces for a too long period, because they can be very verbose, they can "pollute" the log files and moreover could impact performance of FAS. So, do not forget to set back the log level to the standard INFO level.

# 6.3 Monitoring FAS alarm / raw event export application

## 6.3.1 Check database connection

You can check the connection to your FAS datamart with the following command:

```
vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02}  -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02}
```

If you get the SQL prompt, it means that your configuration of FAS has a valid connection to your FAS datamart; else, it might mean that your FAS database is not running, or if it is running it means that you might not have a correct FAS configuration; in that case, please review your *ossa_fault.cfg* configuration file and reconfigure FAS if needed.

## 6.3.2 Check data in database

Some SQL scripts are available in order to get some figures on the data stored in datawarehouse.

For example, by executing the following command, you can check the number of records for each FAS datamart table:

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -f ${OSSA_HOME}/misc/fault_count.sql
```

Here is an example of result:

| Table | Count |
|---|---|
| FCT_FAULT | 5,184,183 |
| FCT_FAULT_CURRENT | 6,065,749 |
| FCT_FAULT_COMMENT | 28,049,774 |
| DIM_ALARMTYPE | 10 |
| DIM_DOMAIN | 10 |
| DIM_MANAGEDOBJECT | 9,540 |
| DIM_OPERATIONCONTEXT | 11 |
| DIM_PROBABLECAUSE | 149 |
| DIM_PROBLEMSTATUS | 3 |
| DIM_SEVERITY | 6 |
| DIM_SPECIFICPROBLEMS | 0 |
| DIM_PROPOSEDREPAIRACTIONS | 0 |
| DIM_STATE | 4 |
| DIM_USER | 27 |

You can also get some statistics on the rate of population of alarm data in the database.

The following command give you the number of records inserted into FCT_FAULT per minute during the last hours:

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -f ${OSSA_HOME}/misc/ff_count_by_minute.sql
```

Here is an example of result:

| Time | Count |
|---|---|
| 2016-03-11 16:04:00 | 8000 |
| 2016-03-11 16:03:00 | 2000 |
| 2016-03-11 16:02:00 | 2016 |
| 2016-03-11 16:01:00 | 1398 |
| 2016-03-11 16:00:00 | 2532 |

You can also check the number of records inserted into FCT_FAULT per hour for each Operation Context:

```
$ vsql -h ${OSSA_DB_HOST_02} -d ${OSSA_DB_NAME_02} -U ${OSSA_DB_USER_02} -w
${OSSA_DB_PASSWORD_02} -f ${OSSA_HOME}/misc/ff_count_by_hour_by_oc.sql
```

Here is an example of result:

| TIMESTAMP_TRUNC | operationcontextname | count |
|---|---|---|
| 1/30/2015 13:00 | ossv040_ns:.oper1 | 19,357 |
| 1/30/2015 13:00 | ossv040_ns:.oper2 | 19,163 |
| 1/30/2015 14:00 | ossv040_ns:.oper1 | 19,409 |
| 1/30/2015 14:00 | ossv040_ns:.oper2 | 19,555 |
| 1/30/2015 15:00 | ossv040_ns:.oper1 | 19,904 |
| 1/30/2015 15:00 | ossv040_ns:.oper2 | 19,523 |
| 1/30/2015 16:00 | ossv040_ns:.oper1 | 19,500 |
| 1/30/2015 16:00 | ossv040_ns:.oper2 | 19,226 |

## 6.4 Kafka admin tools

Please refer to the chapter "*Starting and stopping kafka*" from the *HPE Unified Mediation Bus - Installation and Configuration Guide*.

## 6.5 Tools for finding alarms in Kafka

When troubleshooting some exceptions, you might need to check the input alarm messages.
The *kafka_get_alarms.sh* tool can help you to search for alarms within Kafka:

```
$ kafka get alarms.sh --help

Usage: kafka_get_alarms.sh [-options] <OCName> <AOID> <from-date> <to-date>
       kafka get alarms.sh [-options] <OCName> <AOID> <Last Modification Time>
       kafka get alarms.sh [-options] <partition> <offset>
       kafka get alarms.sh [-options] <from-date> <to-date>

where options include:
       --zookeeper-connect      [mandatory] set zookeeper link
       --topic                  set topic property, default 'TeMIP_AO_Event'
       --partitions             set partitions number, default '64'
       --port                   set kafka port, default '9092'

e.g.
kafka_get_alarms.sh --zookeeper-connect myhost1.com,myhost2.com my_ns:.oper1 2433820 2015-01-
24_06:00:00 2015-01-25_12:00:00

kafka get alarms.sh --zookeeper-connect myvm1.com my ns:.SysTest.AH.myvm1.Endu.ISTEndu.OC IST
52012761 2015-01-27 17:44:52
```

```
kafka get alarms.sh --zookeeper-connect myvm1.com 8 5001234

kafka_get_alarms.sh --zookeeper-connect myvm1.com 2015-01-25_07:05:00 2015-01-25_07:06:00
```

# 6.6 Monitoring execution of default summarizations

## 6.6.1 Batch job monitoring console

You can monitor the execution of the summarizations batch jobs within the batch job monitoring console, available at:

```
http://<OSSA_Server>:8080/#/batch
```

where you can see the list of batch jobs loaded, the history of executions, and the history of detailed steps of each job execution.

## 6.6.2 Logs

Within the server logs, you can see some information about the executions of summarizations:

```
grep -i  com.hp.ossa.batch.batchlet.summ server.log
```

You will find:
- the number of calculation periods summarized per summarization
- the completion status of each summarization
- some internal processing durations, and number of rows taken as input, produced as output
- possible errors if any

# 6.7 Specific troubleshooting

## 6.7.1 Kafka message skipped: invalid alarm

If you get this kind of message in FAS logs:

```
This alarm message will be skipped due to missing mandatory attribute[ACKFLAG], alarm id is:
451278, ocname is: ossa.testing
```

It means that the alarm message consumed by FAS is invalid.
Note that in this trace, you get the operation context name (*ocname*) and the identifier of the alarm (*alarm id*).

This will help you in finding the root cause of the problem related to this alarm.

You can get the whole alarm message, by setting log level to DEBUG.

But, you can also use the tool *kafka_get_alarms.sh* described before, and this will give you the kafka alarm message on which you must investigate, for example:

{
"OCName": "ossa.testing",
"Identifier": 1,
"AttributeList": [
{"AttributeId": 1, "AttributeName": "Alarm Type", "AttributeType": 10, "IntValue": [], "LongValue": [4], "StringValue": ["EquipmentAlarm"], "BooleanValue": [], "DoubleValue": []},
...
}