HP OSS Fault Analytics & Statistics

Version 1.1

Installation, Configuration and Administration Guide

Edition: 1.0

For Linux (RHEL 6.5)

October 2015

Legal Notices

Warranty

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

License Requirement and U.S. Government Legend

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

Copyright Notices

Trademark Notices

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

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.

HP Vertica™, the HP Vertica Analytics Platform™ are trademarks of Hewlett-Packard

Firefox® is a registered trademark of the Mozilla Foundation.

Google Chrome® is a trademark of Google Inc.

UNIX® is a registered trademark of The Open Group.

Red Hat® is a registered trademark of the Red Hat Company.

Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

Contents

Figures

Tables

Preface

This document describes information related to the HP OSS Fault Analytics & Statistics V1.1 product.

Product Name: HP OSS Fault Analytics & Statistics

Product Version: 1.1.0

Product Kit Name : ossa-fault-1.1.0-MR.noarch.rpm

Please read this document before installing or using this Software.

Intended Audience

Here are some recommendations based on possible reader profiles:

- Administrators
- 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
HP OSS Fault Analytics & Statistics 1.1.0	• Red Hat Enterprise Linux Server release RHEL 6.5
HP OSS Analytics Foundation 1.1.1	• Red Hat Enterprise Linux Server release RHEL 6.5
HP Vertica Version 7.1	• Red Hat Enterprise Linux Server release RHEL 6.5
HP UMB Server Version 1.0	• Red Hat Enterprise Linux Server release RHEL 6.5
HP Unified OSS Console 2.1	• Red Hat Enterprise Linux Server release RHEL 6.5
HP TeMIP 6.2	• Red Hat Enterprise Linux Server release RHEL 6.5

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

- HP OSS Analytics Foundation Release Notes
- HP OSS Analytics Foundation Installation Configuration and Administration guide
- HP TeMIP Patch readme documents
- HP Unified Mediation Bus - Installation and Configuration Guide
- Kafka doc http://kafka.apache.org/documentation.html

Support

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

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

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

Chapter 1

Product overview

HP OSS Fault Analytics & Statistics overview

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

OSS FAS is based on HP 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 received from HP TeMIP into meaningful information
- Use of the HP 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 information data about alarms (aggregation based on different time granularities and different dimensions)
- Optional default FAS reports (built with Unified OSS Console) based on those summarized tables

Architecture

HP OSS Fault Analytics and Statistics (FAS) solution is composed of three software components:

- TeMIP Analytics
- OSS Analytics foundation
- OSS FAS Core

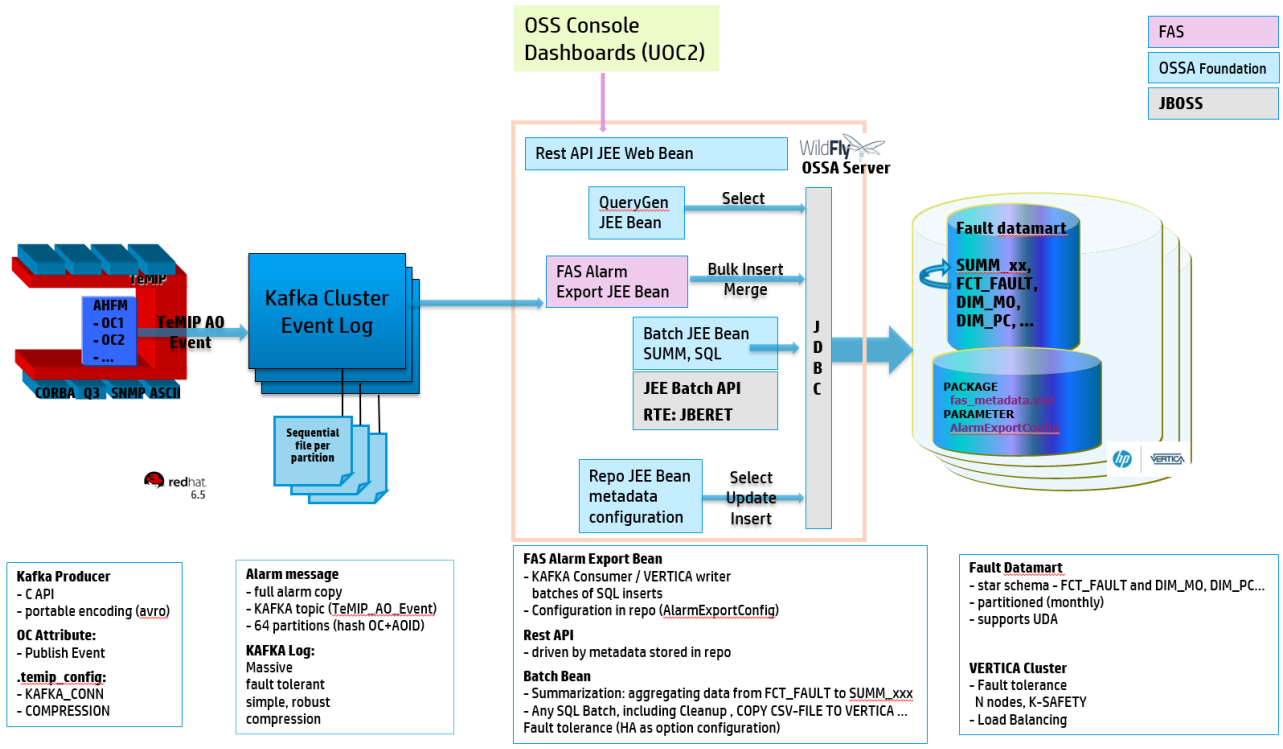

Figure 1 – OSS Fault Analytics & Statistics architecture

Chapter 2

Product Installation and Upgrade

Product Components Introduction

The three software components that compose a HP OSS Fault Analytics & Statistics solution are:

- TeMIP Analytics: delivered through TeMIP patches on top of TeMIP Framework Server V6.2 on RHEL; it is a TeMIP Alarm Handling patch.
- OSS Analytics Foundation product
- OSS FAS core, contains the alarm export application, default summarizations and default UOCV2.x reports on top of default summarizations

Before installation of the **FAS core**, please install **TeMIP Analytics Patch** and the **OSS Analytics Foundation** components.
Note that TeMIP is on an independent platform.

TeMIP Analytics Patch

TeMIP Analytics patches must be installed on the TeMIP Framework Server 6.2 on RHEL platform in order to export alarms in real-time to the Kafka broker.
Those exported alarms into Kafka will then be consumed by the FAS application.

HP OSS Analytics Server

The HP OSS Analytics server is a software component that is a prerequisite for the installation of a FAS solution.
The OSSA Server allows to access and do analysis on the alarms data stored in the Vertica database, thanks to the FAS TeMIP Fault metadata loaded into it.
OSSA server provides a Rest Api for external application to access the data. It brings also the JEE application server (Wildfly/JBOSS8).

Note

For more details, please refer to HP OSS Analytics Foundation documentation.

FAS core

FAS core is a software product. FAS contains the alarm export application which receives and transforms vast amounts of alarm data received from HP TeMIP component and populates a 'fault' datawarehouse based on HP Vertica. This mediation layer allows to collect fault information in real time.
FAS core also contains default summarizations: this is a batch job to be optionally deployed and scheduled into the OSSA Foundation Server, if one wants standard default summarizations to be executed.

FAS core also contains standard default reports to display data produced by default summarizations. Default reports are based on HP Unified OSS Console V2.x views and workspaces.

Prerequisites

Hardware and Operating Systems prerequisites

The HP FAS is supported on **Red Hat Enterprise Linux v6.x (x86-64)**.

Before installing OSS Analytics server, verify that your system meets the following minimum requirements.

Requirements	Minimal
CPU	1 CPU 2.5 GHz 4 cores
RAM	8 GB
Hard disk Size	50 GB
Network	100 MB Ethernet

Table 2 – Minimal hardware requirements for OSS FAS on *Linux*

For a real sizing exercise, please contact the FAS Product Manager.

Software Prerequisites

Operating System

HP FAS is supported on Red Hat Enterprise Linux Server release 6.x **(x86-64)** .

TeMIP Framework Server V6.2 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 real-time to the Kafka broker:

TFRPATCH12183TEST-MSLV62L – AHFM Dictionary Patch

*TFRPATCH12183TESTMV62*L – AHFM Library Patch

TEMIPTFRLIN_00210 – TFRTFC Patch

Note

For more details, please refer to individual Patch readme
TFRPATCH12183TESTNV62L.text (or superseding one)

Kafka / Zookeeper Server

Kafka / Zookeeper broker are prerequisites for FAS:

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

Java

Please refer to the “OSS Analytics Foundation Installation Guide”.

OSS Analytics Foundation

OSS Analytics Foundation is a prerequisite for FAS. So, please refer to the “OSS Analytics Foundation Installation Guide”.

HP VERTICA Database

FAS stores domain specific data into HP Vertica datawarehouse. Thus, it requires the HP Vertica 7.1.x database software to be installed.

The vertica client should be ready when installing the OSS Analytics Foundation.

Please refer to OSS Analytics Foundation documentation for more information.

Code Signing

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.

Check whether gnupg gpg is installed on the system. If no, install gnupg gpg

Configure GPG for accepting HP signature. The steps are the following:

Log as root on your system

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.

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.*

Install TeMIP Analytics Patch

The patches mentioned in Section TeMIP Analytics Patch must be installed on all TeMIP servers that host Operation Contexts.
(For more details refer to corresponding patch text)

- Login as root user
- Stop TeMIP
- Change directory to Patch kit.
- Install all the patches as shown below.

```
# pkgadd -a `pwd`/adminFile -R /usr/opt/TeMIP<xxx> \
    -d `pwd` <EPatch_Name>
```

Where `/usr/opt/TeMIP<xxx>` is TeMIP release tree.

Ex : `pkgadd -a `pwd`/adminFile -R /usr/opt/TeMIP-V62L \`
`-d `pwd` TFRPATCH12183TESTGV62L`

- Activate the patch

```
# /usr/opt/TeMIP-V62L/temip/bin/temip_activate
```

Refer to corresponding patch text for post installation procedures.

And refer to the UMB documentation for the installation of UMB-Kafka / UMB-Zookeeper broker components (see *HP Unified Mediation Bus - Installation and Configuration Guide*).

WARNING:

In addition to the configuration of Kafka / Zookeeper done while installing the umb zookeeper/kafka 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 x 64 / alarm size / alarm rate should give you an approximation of the history time window size of the alarms retained)

Install OSS Analytics Foundation

Please follow the instructions given in the HP OSS Analytics Foundation Installation guide.

Install the FAS Core

Unix user

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

Install FAS Core kit

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

```
# rpm -Uvh ossa-fault-1.1.0-MR.noarch.rpm
```

This OSS FAS core kit is then installed onto the */opt/ossa* directory.

FAS core environment variables setting

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.
For example:

```
OSSA_DB_TYPE_02=vertica
OSSA_DB_HOST_02=<host_name>
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=128
```

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

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

Create FAS core database user

When creating the database user, a dedicated resource pool will also be created.

```
$ vsql -d ${OSSA DB NAME 02} -h ${OSSA DB HOST 02} -U dbadmin \
     -w <dbadminpwd> -f ${OSSA_HOME}/ddl/create_user_fas.sql
```

Create FAS core database schema

```
$ 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} -v ossa_user=${OSSA_DB_USER_01} \
   -f ${OSSA_HOME}/ddl/create_schema_fault.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.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
```

Note

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

```
$ 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_views.sql
```

Configuring FAS alarm export application

Here are the main parameters that can be defined for FAS alarm export application:

Property	Sub_Property	Example value	Description
consumerGroupConfigs			
	group.id	myproject-cg-kafka_south-fas_north-01	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 group. One definite alarm will be consumed only by one consumer under the group. And one alarm will be repeatly consumed by several processes by setting different group id for each process. Recommended naming convention, <project>-cg-<from-kafka-cluster-name>-<to-consumer-name>. This is useful for troubleshooting in case several FAS application consumes alarm messages from several kafka clusters.
	zookeeper.connect	localhost:2181	Specifies the ZooKeeper connection string in the form hostname:port where host and port are the host and port of a ZooKeeper server. If there are several ZooKeeper nodes for insurance, just use the form hostname1:port1,hostname2:port2.
	auto.commit.enable	false	This one is enforced to false by FAS alarm export application whatever the value specified in config, it will not be taken into account. The offset must be committed by FAS alarm export application explicitly and never auto committed by Kafka.

Property	Sub_Property	Example value	Description
	auto.offset.reset	smallest	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 consumer Default value should be smallest.
consumerThreadNumber		16	
writerConfig	batchSize	1000	As soon as thread has accumulated batch size alarms, then the batch is flushed to DB.
	writeIntervalSeconsd	60	flush occurs always after this delay even if less that batch size alarms are not been accumulated.
commitOffsetsIntervalMinutes		5	Default 5 minutes. Configure how often to open the offset commit window. To avoid data loss, we have to control the offset commit in FAS program. We cannot depend on Kafka's auto commit.
temipClassSpecificProblems			Some access modules and associated tTeMIP classes use specific problems in degenaretd way to encode something else than specific problems, e.g. notification id. In such case the mapping of specific problems should be avoided.
	temipClass		type temipClass which you want to configure for specific problem mapping.
	supportsSP		Default false. Decide whether the Specific Problems attribute related to the temipClass above should persist to DIM_SPECIFICPROBLEMS

How to set those parameters ? Here is an example:

```
$ ${OSSA HOME}/bin/ossa-repo.sh setParam OSSA FAULT AlarmExportConfig '
{ "consumerGroupConfigs" : [ { "consumerGroupProperties" : { "group.id"
: "ossa-cg-kafka_south-fas_north-01", "zookeeper.connect" :
"localhost:2181", "auto.commit.enable" : "true", "auto.offset.reset" :
"smallest", "rebalance.max.retries" : "20", "rebalance.backoff.ms" :
"10000", "zookeeper.session.timeout.ms" : "60000",
"zookeeper.connection.timeout.ms" : "60000" }, "consumerThreadNumber" :
1 } ] , "temipClassSpecificProblems" : [ {"temipClass" : "OSI SYSTEM",
"supportsSP" : true }, {"temipClass" : "", "supportsSP" : false }] ,
"writerConfig" : { "batchSize" : 10000, "writeIntervalMaxSeconds" : 60
} } '
```

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

Configure default summarizations

Please follow the steps below in order to configure the standard default summarizations.

Note

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

In the terminal where you sourced *ossa_env.sh*, as *ossa* user,
- load the description of the processing of the summarization job
- then, load the execution description of the summarization job
(by default, the scheduling defines an execution every 10 mn, see the .json file)

```
$ source ${OSSA HOME}/bin/ossa env.sh

$  sh ${OSSA_HOME}/bin/ossa-repo.sh loadParam FAS FASsummJob.xml \
          ${OSSA HOME}/repo-fas/FAS/batch/FASsummJob.xml

$ sh ${OSSA_HOME}/bin/ossa-repo.sh loadParam FAS BATCH_FASsumm.json \
          ${OSSA HOME}/repo-fas/FAS/batch/BATCH FASsumm.json
```

Once the summarizations batch job description is loaded, the summarizations will be automatically run regularly according to the schedule.

Deploy FAS alarm export application

```
$ source ${OSSA_HOME}/bin/ossa_env.sh

$ sh ${OSSA_HOME}/bin/ossa_config_fault.sh
```

Load default standard FAS reports for HP Unified OSS Console

In the terminal where you sourced *ossa_env.sh*, as *ossa* user, load the default standard FAS reports (views and workspaces) in OSS Analytics Server repository, for HP Unified OSS Console.

```
$ source ${OSSA_HOME}/bin/ossa_env.sh

$ cd /opt/ossa/repo-fas/FAS/

$ ${OSSA_HOME}/bin/ossa-repo.sh loadMetadataViewsWks \
      ${OSSA HOME}/repo-fas/FAS/metadata/ossa fault metadata.xml \
      ${OSSA_HOME}/repo-fas/FAS/ui/views.json \
      ${OSSA_HOME}/repo-fas/FAS/ui/workspaces.json
```

Once this command is run, you will be able to view the reports in the HP UOC Console workspaces: *FAS Alarm Health Reports*, *FAS Network Health Reports*, *FAS Network Management Health Reports*.

Verify the installation

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

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

The output should display the *ossa-server* and the *ossa-fault* packages installed.

You can run the following command in order to check that the JBoss 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 see the *ossa-restapi* and *ossa-fault* ear.

TCP ports used

The following **TCP ports** are 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 OSSA REST Server
- *Vertica documentation* for Vertica
- *Unified OSS Console Installation & Configuration Guide* for UOC

Uninstall the OSS Fault Analytics & Statistics

```
$ source ${OSSA HOME}/bin/ossa env.sh
$ jbossstop
```

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

```
#  rpm -e ossa-fault-1.1.0-MR.noarch
```

Uninstall TeMIP Analytics Patches

- Login as root user
- Stop TeMIP
- Remove individual patch as shown below.

```
# pkgrm -R /usr/opt/TeMIP<xxx> <EPatch Name>

Where /usr/opt/TeMIP<xxx> is TeMIP release tree.
Ex: pkgadd -R /usr/opt/TeMIP-V62L TFRPATCH12183TESTGV62L
```

Chapter 3

Product Administration

Configure and Administrate TeMIP Analytics Patch

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

- Create topic with right way

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

`Replication-factor` should be configured with a number and must be less than or equal to the number of Kafka hosts.

Recommended number of partitions in Kafka is 64.

Kafka Broker must be configured with name '**TeMIP_AO_Event**'.

- Edit `/var/opt/temip/conf/.temip_config` and add the Kafka broker information.

```
TEMIP AH KAFKA CON STRING=host1:port1,host2:port2
ENABLE_KAFKA_COMPRESSION=ON
```

Ex: `TEMIP_AH_KAFKA_CON_STRING=myhost1.com:9092,myhost2.com:9092`

Where `port1`, `port2` are Kafka Broker ports and `host1`, `host2` are the hosts where Kafka Broker is running.

- Restart Alarm Handling FM.

```
# manage restart mcc 0 application alarm_handling_fm
```

- Enable operation contexts to publish events to kafka.

```
# manage set operation context <OC Name> Publish Events=True
```

Administrate OSS Analytics Server

Please follow the *HP OSS Analytics Foundation Installation and Administration guide*.

Administrate FAS Core

Stop the FAS core

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

```
$ jbossstop
```

This will stop the whole OSS Analytics Server (including FAS).

Start the FAS core

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

```
$ jbossstart
```

This will start the whole OSS Analytics Server (including FAS).

Chapter 4

Product tuning

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

JBoss

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).

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 OSS FAS Core parameters*)
- parallel connections for FAS summarizations (see parameter *`nbThreads`* in summarizations batch job description `FASsummJob.xml` for each of the three domains run concurrently : 3 x 10 threads by default)
So, if *`consumerThreadNumber`* = *4* and summarizations *`nbThreads`* = *10,* *`OSSA_DS_MAX_POOL_SIZE_02`* must be equals to 34 (4 + 3 x 10).
Note that this number must also correspond to the Vertica FAS resource pool parameter `plannedconcurrency` (see sections below)

Reminder: when modifying `ossa.cfg` the OSSA server must be re-configured, so please, stop OSSA server, and execute: `${OSSA_HOME}/bin/ossa_config.sh`
The same remark is valid for FAS: in case you modify the `ossa_fault.cfg`, FAS must be re-configured, so please execute: `${OSSA_HOME}/bin/ossa_config_fault.sh`

Vertica

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

Here are the other recommendations concerning resource pools:

- The Vertica **tuple mover** resource pool named "***tm***" must be the only one with HIGH *`runtimepriority`*.

- Thus, your user defined resource pools 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.
  Define also 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 OSS Analytics datasource maximum pool size*** / ***JBoss 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 for example.

For more detailed information about resource pool settings you can refer to Vertica documentation:
http://my.vertica.com/docs/7.1.x/HTML/index.htm#Authoring/AdministratorsGuide/ResourceManager/GuidelinesForSettingPoolParameters.htm

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 as *dbadmin*:

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

On client side, this can be interesting to set load balancing for requests coming from OSSA REST Server where there could be a high number of parallel requests. For doing this, please modify the OSSA 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_NAME_01}?BackupServerNode=<my2ndVerticaNode>,
<my3rdVerticaNode>&ConnectionLoadBalance=1"
```

Chapter 5

Troubleshooting

Monitoring FAS Alarm export application

Check Data in Database

There're some SQL scripts for checking alarm data in database. For example, for checking the number of records of all the fact and dimension tables.

```
$ 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:

TBL	CNT
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

For checking the number of records inserted into FCT_FAULT during a short period:

```
$ 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
2/2/2015 10:05	1891
2/2/2015 10:00	2038
2/2/2015 9:55	1907
2/2/2015 9:50	2364

2/2/2015 9:45	2015
2/2/2015 9:40	2439
2/2/2015 9:35	1897

For checking the number of records inserted into FCT_FAULT for each Operation Context per hour:

```
$ 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

Check CPU and Memory

You can check the CPU usage with the following commands:

```
vmstat
```

```
top
```

You can check the memory usage with the following commands:

```
free -t -m
```

```
cat /proc/meminfo
```

Check Disk Space

You can check the disk space usage with the following commands:

```
df
```

which displays information on free disk space,

```
du
```

which displays summary of disk usage.

Check Database Connection

You can check database connection by command,.

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

If you can successfully login, that means database connection is ok. Otherwise, there's something wrong with the database or network.

Monitoring execution of default summarizations

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.

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
```

- the number of calculation periods summarized per summarization
- the completion status of each summarization
- possible errors if any

General Troubleshooting

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.

Log Level

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

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

```
$ 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
```

When exceptions occur, you can set level name to `DEBUG` to get more useful information.

To enable logging of alarm message, please set level name to `TRACE`.

Once investigation is completed, do not forget to set back the log level to the standard level `INFO`, in order to avoid polluting the log files and to avoid impacting the performance of FAS:

```
$ ossa_fault_set_log INFO
```

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 as below to get 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
```

Trace mask will be reset to previous value when the process is restarted.

Kafka Admin Tools

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

Tools for finding alarms

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

```
$ 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
```

Specific Troubleshooting

Invalid Alarm

When there are invalid alarms found, please check log file to find out the `corresponding operation context name (`*`OCName`*`) and identifier of the alarm`.

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

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

You can also use the tool `kafka_get_alarms.sh` described above for finding the alarm.

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