ERICSSON

Ericsson Catalog Manager and Ericsson Order Care

Realize Higher Consistency for Faster Time-to-Revenue

Execution Environment - Fault Management, Logging, and Performance Management

Contents

1 Introduction

The AVM supports fault management (FM), logging, and performance management (PM) capabilities for Execution Environment. Execution Environment is a framework for realizing common functionality in BUSS products. It supports a modular way of building BSS/OSS solutions, while making sure that the BSS/OSS offerings look alike when it comes to external interfaces and characteristics.

This document describes how the AVM supports FM, logging, and PM in Execution Environment.

1.1 FM and Logging in Execution Environment

Execution Environment provides an API for domain applications to call to log faults into a file as well as sending faults as alarms to the OSS side. For logging faults into a file, Execution Environment uses logback to write faults to /var/log/ngee/ngee.log file. For alarms, Execution Environment uses JavaOaM FM framework and Ericsson SNMP Agent (ESA).

1.1.1 Logging in Execution Environment

Logging in Execution Environment is implemented using the following:

- SLF4J API
- Logback
- Back-end to collect and view logs

Execution Environment provides the following for logging:

- APIs built into Domain Server and OSGi Container
- Default configuration
- Mechanism to view logs (from NGMS)

For logging, domains on Execution Environment can:

- Use the APIs to write traces/logs
- Change the default configuration to suit
- View logs using the supplied mechanism

1.1.2 FM in Execution Environment

FM in Execution Environment is implemented using the following:

- Notification2Alarm interface
  (configuration decides to send notification or raise alarm)
- JavaOaM interface
- ESA Adapter
- ESA
- Additional functionality from CRS and Execution Environment

Execution Environment provides the following for FM:

- ECIM Alarm Model and generated configuration for EE alarms
- Unique ECIM-registered alarm IDs with Alarm Component Name in Execution Environment
- Generated documentation from ECIM Models
- Framework for Domains to model and configure their own notifications and alarms

2 Fault Management

2.1 Implementation in AVM

2.1.1 APIs

Two new scriptable APIs, Global.raiseAlarm() and Global.clearAlarm(), are now available and have been implemented in the CwfScriptGlobalFunctions class.

2.1.2 Alarm Logger Class

A new class “CwfAlarmLogger” is created in the com.conceptwave.system package.

“CwfAlarmLogger” has raiseAlarm() and clearAlarm() methods which can be called from scriptable API and AVM platform.

“CwfAlarmLogger” sends a fault event to raise/clear an alarm to FM when AVM is running on Execution Environment, and sends notification to JMX console "CWAVMAlarm" MBean under "com.conceptwave.AVM".

2.1.3 MBean

A new "CWAVMAlarm" MBean has been added under "com.conceptwave.AVM." It is a stateful alarm bean with “Number of active alarms” attribute, and “Get Active Alarms” operation and notifications.

"CWAVMAlarm" MBean maintains stateful (meaning that once it is raised, it stays “ON” until it is cleared) active alarms. Notifications are based on one time raise/clear events (instead of periodic alarm notifications when it’s active and no notification when the alarm is cleared).

2.1.4 Alarm: Database is down or unreachable

The *Database is down or unreachable* alarm has been implemented for Execution Environment. Additional alarms are expected to be identified and implemented later.

2.1.5 Alarm Trigger

The *Database is down or unreachable* alarm is implemented by making use of CwfAVMHeartBeat class. Whenever the CwfAVMHeartBeat class enters or exits the *Lost DB Connection Mode*, it will trigger the raising or clearing of the *Database is down or unreachable* alarm.

2.1.6 Alarm Not Triggered or Cleared

This section lists situations where an alarm is not raised or is cleared, to meet the requirements of Execution Environment.

- When the AVM is being shut down, no “Database is down or unreachable” alarm will be raised (that is, CwfAVMHeartBeat thread will exit without raising alarm).
- When the AVM is being shut down and the “Database is down or unreachable” was raised, CwfAVMHeartBeat will clear the alarm.
- When the AVM is being shut down, no alarm is raised to indicate that AVM is shutting down.

2.2 Alarm: AVM, Database is down or unreachable

2.2.1 Define a New Alarm

A new (Ericsson Common Information Model) ECIM FM alarm model is defined in the **AvmFmInstances_mp.xml** file. This file is found under …/opt/ngee/latest/base/conf on Execution Environment.

The following code shows the *AVM, Database is down or unreachable* alarm being defined in the **AvmFmInstances_mp.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?><!--
<models>
    <dtdVersion>E1</dtdVersion>
    <momMetaData>

    <mib name="AvmFmInstances" release="1" version="0">
        <object parentDn="">
         <hasClass name="Fm"> <mimName>ECIM_FM</mimName> </hasClass>
         <slot name="fmId"> <value>1</value> </slot>
        </object>
        <object parentDn="Fm=1">
         <hasClass name="FmAlarmModel"> <mimName>ECIM_FM</mimName>
</hasClass>
         <slot name="fmAlarmModelId">
                     <value>Avm</value>
         </slot>
        </object>
        <object parentDn="Fm=1,FmAlarmModel=Avm">
         <hasClass name="FmAlarmType"> <mimName>ECIM_FM</mimName>
</hasClass>
         <slot name="fmAlarmTypeId"> <value>AvmDatabaseDown</value>
</slot>
         <slot name="eventType"> <value>QUALITYOFSERVICEALARM</value>
</slot>
         <slot name="majorType"> <value>193</value> </slot>
         <slot name="minorType"> <value>1377000</value> </slot>
         <slot name="specificProblem">
                     <value>AVM, Database is down or unreachable
</value>
         </slot>
         <slot name="probableCause"> <value>612</value> </slot>
         <slot name="isStateful"> <value>true</value> </slot>
         <slot name="additionalText">
                     <value>AVM, Database is down or unreachable
</value>
         </slot>
         <slot name="defaultSeverity"> <value>CRITICAL</value> </slot>
        </object>
    </mib>
</models>
```

2.2.2 Add a New Error Code

A new error code is added by creating a new **AvmAlarmsDefs.xml** file (under /opt/ngee/latest/base/conf) and defining the code in that file. The error code has to be an even number. The following shows **AvmAlarmsDefs.xml** file being created for the *AVM, Database is down or unreachable* alarm:

```
        <!-- AVM range is 1377000-1379998 -->
        <alarmSpecification active="yes">
                <moduleId>AVM</moduleId>
                <errorCode>1377000</errorCode>
                <severity>3</severity> <!--critical-->
                <modelDescription>AVM, Database is down or
unreachable</modelDescription>
                <activeDescription>AVM, Database is down or
unreachable</activeDescription>
                <eventType>3</eventType> <!-- Quality of Service -->
                <probableCause>612</probableCause> <!-- Out of Service
-->
        </alarmSpecification>
```

2.2.3 Add New Notification IDs for the Alarm

New notification IDs for the alarm need to be added using the **NgeeInstances_mp.xml** file, which is found in the …/opt/ngee/latest/base/conf folder. One notification ID needs to be added for RAISE_ALARM and another notification ID needs to be added for CLEAR_ALARM. The following code shows notification IDs being added for the *AVM, Database is down or unreachable* alarm.

```
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification">
<mimName>NgeeMom</mimName> </hasClass>
            <slot name="notificationId"> <value>1377000</value>
</slot>
            <slot name="notificationDesc">
                <value>AVM, Database is down or unreachable</value>
            </slot>
            <slot name="notificationType"> <value>RAISE_ALARM</value>
</slot>
            <slot name="logLevel"> <value>ERROR</value> </slot>
            <slot name="errorCode"> <value>1377000</value> </slot>
            <slot name="moduleId"> <value>AVM</value> </slot>
        </object>
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification">
<mimName>NgeeMom</mimName> </hasClass>
            <slot name="notificationId"> <value>1377001</value>
</slot>
            <slot name="notificationDesc">
                <value>AVM, Database is down or unreachable</value>
            </slot>
            <slot name="notificationType"> <value>CLEAR_ALARM</value>
</slot>
            <slot name="logLevel"> <value>INFO</value> </slot>
            <slot name="errorCode"> <value>1377000</value> </slot>
            <slot name="moduleId"> <value>AVM</value> </slot>
        </object>
```

2.3 Add a New Alarm, and Raise and Clear an Alarm

1. Define a new alarm in the AvmFmInstances_mp.xml file.
2. Add a new error code (even number) in the AvmAlarmsDefs.xml file.
3. Add new notification IDs for the alarm in Step 1:

   a. One for RAISE_ALARM - same as error code in Step 2
   b. One for CLEAR_ALARM - error code in Step 2 + 1

4. To raise the alarm after detecting fault condition, do one of the following:
   a. Call CwfAlarmLogger.*getInstance*().raiseAlarm() from platform Java code
   b. Call Global.raiseAlarm() from Javascript

5. To clear the alarm after detecting the fault condition is cleared, do one of the following:
   a. Call CwfAlarmLogger.*getInstance*().clearAlarm() from platform Java code
   b. Call Global.clearAlarm() from Javascript

3 Logging

Current logging on Execution Environment is based on SLF4J API and logback.

Each Execution Environment OaM server component has its own logback configuration and log file. For example, the oam-agent daemon has **oamLogback.xml** configuration file (under /opt/ngee/latest/base/conf) and uses the **ngeeOamAgent.log** file (under /var/log/ngee). The Java VM argument used by oam-agent is:

-Dlogback.configurationFile=$NGEE_BASE/conf/oamLogback.xml

For application and Java Execution Environment OaM, the client library should be using the following **ngeeLogback.xml** configuration and **ngee.${HOSTNAME}.log** log file (under /var/log/ngee).

The following is the section related to “NGEE-APPS”:

```
<appender name="NGEE-APPS"
   class="ch.qos.logback.core.rolling.RollingFileAppender">
   <file>/var/log/ngee/ngee.${HOSTNAME}.log</file>
~~~
~~~
   <encoder>
           <pattern>%date %level [%thread] %logger{10} [%file:%line] %msg%n
           </pattern>
   </encoder>
</appender>
~~~
~~~
<root level="ERROR">
   <appender-ref ref="NGEE-APPS" />
</root>
~~~
~~~
```

3.1 AVM Logging on Execution Environment

AVM running on Execution Environment requires the following changes from the regular AVM logging.

3.1.1 Create a New Template

Create the following new template: **logback-ngee-template.xml**

The new template should be a merged version of **logback-container-template.xml** and **ngeeLogback.xml**. It should have a “$REPLACE$” section so that the list of loggers and associated levels can be configured. The log file should be under /var/log/ngee

3.1.2 Add a New Execution Environment logging mode

Detect if AVM is running on Execution Environment and use the **logback-ngee-template.xml** file.

Detecting if AVM is running on Execution Environment is based on the following code.

```
import com.ericsson.bss.oam.configuration.Configuration;
~~~
isAvmOnNgee = !Configuration.getInstance().isEmpty()
```

4 Example of How Execution Environment FM APIs and Associated Configurations Work

4.1 Execution Environment OaM FM API Interface

```
NotificationLogger logger =

NotificationLoggerFactory.getInstance(NotificationLogger4Ngee.c
lass);
logger.postNotification(NgeeFmTest.class.getName(), 1,
"1.1.1.100.5", "Error message");

// Parameters of the above logger.postNotification()
// loggingClassName: name of the class to log the faults
//                   (usually the application class name)
// notificationID:   id that matches to the one defined in the
ECIM model
// resourceID:       SNMP OID (e.g. 1.1.1.100.1, 1.1.1.100.2,
etc.).
//                   uniquely identify your cluster, process,
etc.
```

```
// message:          the fault message
```

4.2 Alarm Configuration

The above API is based on the Notification2Alarm interface (configuration decides to send notification or raise alarm).

Configurations for FM Alarm consists of the following:

- ECIM Alarm Model: ECIM FM Instance
- ESA alarm configuration file

4.2.1 Example of ECIM Instance

```
<?xml version="1.0" encoding="UTF-8"?><!---
<models>
    <dtdVersion>E1</dtdVersion>
    <momMetaData>

    <mib name="NgeeInstances" release="0" version="1">
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification">
                <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="notificationId"> <value>7</value> </slot>
            <slot name="notificationDesc">
                <value>Service cannot be restarted.</value>
            </slot>
            <slot name="moduleId"> <value>NGEE.SM</value> </slot>
            <slot name="notificationType"> <value>RAISE_ALARM</value> </slot>
            <slot name="logLevel"> <value>ERROR</value> </slot>
            <slot name="errorCode"> <value>721001</value> </slot>
        </object>
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification">
                <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="notificationId"> <value>5</value> </slot>
            <slot name="notificationDesc"> <value>Service is down.</value> </slot>
            <slot name="moduleId"> <value>NGEE.SM</value> </slot>
            <slot name="notificationType"> <value>RAISE_ALARM</value> </slot>
            <slot name="logLevel"> <value>ERROR</value> </slot>
            <slot name="errorCode"> <value>721000</value> </slot>
        </object>
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification">
                <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="notificationId"> <value>6</value> </slot>
            <slot name="notificationDesc"> <value>Service is down.</value> </slot>
            <slot name="moduleId"> <value>NGEE.SM</value> </slot>
            <slot name="notificationType"> <value>CLEAR_ALARM</value> </slot>
            <slot name="logLevel"> <value>ERROR</value> </slot>
            <slot name="errorCode"> <value>721000</value> </slot>
```

```
</object>
<object parentDn="Ngee=1,NgeeFm=1">
    <hasClass name="NgeeFmNotification">
        <mimName>NgeeMom</mimName>
    </hasClass>
    <slot name="notificationId"> <value>2</value> </slot>
    <slot name="notificationDesc"> <value>Info log.</value> </slot>
    <slot name="notificationType"> <value>NOTIFICATION_ONLY</value> </slot>
    <slot name="logLevel"> <value>INFO</value> </slot>
    <slot name="errorCode"> <value>-1</value> </slot>
</object>
<object parentDn="Ngee=1,NgeeFm=1">
    <hasClass name="NgeeFmNotification">
        <mimName>NgeeMom</mimName>
    </hasClass>
    <slot name="notificationId"> <value>4</value> </slot>
    <slot name="notificationDesc"> <value>Error log.</value> </slot>
    <slot name="notificationType"> <value>NOTIFICATION_ONLY</value> </slot>
    <slot name="logLevel"> <value>ERROR</value> </slot>
    <slot name="errorCode"> <value>-1</value> </slot>
</object>
<object parentDn="">
    <hasClass name="ManagedElement">
        <mimName>ECIM_Top</mimName>
    </hasClass>
    <slot name="managedElementId"> <value>1</value> </slot>
</object>
<object parentDn="">
    <hasClass name="Ngee">
        <mimName>NgeeMom</mimName>
    </hasClass>
    <slot name="ngeeId"> <value>1</value> </slot>
</object>
<object parentDn="Ngee=1,NgeeFm=1">
    <hasClass name="NgeeFmNotification">
        <mimName>NgeeMom</mimName>
    </hasClass>
    <slot name="notificationId"> <value>1</value> </slot>
    <slot name="notificationDesc"> <value>Debug log.</value> </slot>
    <slot name="notificationType"> <value>NOTIFICATION_ONLY</value> </slot>
    <slot name="logLevel"> <value>DEBUG</value> </slot>
    <slot name="errorCode"> <value>-1</value> </slot>
</object>
<object parentDn="Ngee=1,NgeeFm=1">
    <hasClass name="NgeeFmNotification">
        <mimName>NgeeMom</mimName>
    </hasClass>
    <slot name="notificationId"> <value>0</value> </slot>
    <slot name="notificationDesc"> <value>Trace log.</value> </slot>
    <slot name="notificationType"> <value>NOTIFICATION_ONLY</value> </slot>
    <slot name="logLevel"> <value>TRACE</value> </slot>
    <slot name="errorCode"> <value>-1</value> </slot>
</object>
<object parentDn="Ngee=1,NgeeFm=1">
    <hasClass name="NgeeFmNotification">
        <mimName>NgeeMom</mimName>
    </hasClass>
    <slot name="notificationId"> <value>8</value> </slot>
    <slot name="notificationDesc">
        <value>Service cannot be restarted.</value>
    </slot>
    <slot name="notificationType"> <value>CLEAR_ALARM</value> </slot>
    <slot name="moduleId"> <value>NGEE.SM</value> </slot>
    <slot name="logLevel"> <value>ERROR</value> </slot>
    <slot name="errorCode"> <value>721001</value> </slot>
```

```
        </object>
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification">
                <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="notificationId"> <value>3</value> </slot>
            <slot name="notificationDesc"> <value>Warning log.</value> </slot>
            <slot name="notificationType"> <value>NOTIFICATION_ONLY</value> </slot>
            <slot name="logLevel"> <value>WARN</value> </slot>
            <slot name="errorCode"> <value>-1</value> </slot>
        </object>
        <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmEsaAdapter">
                <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="ngeeFmEsaAdapterId"> <value>1</value> </slot>
            <slot name="ipAddr"> <value>localhost</value> </slot>
            <slot name="port"> <value>8666</value> </slot>
        </object>
        <object parentDn="Ngee=1">
            <hasClass name="NgeeFm">
                <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="ngeeFmId">
                <value>1</value>
            </slot>
        </object>
    </mib>
</models>
```

4.2.2 Example of ESA Alarm Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<alarmDefinitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ericsson.com/esa"
xsi:schemaLocation="http://www.ericsson.com/esa fmAlarmDefinition.xsd">

        <!-- SM range is 721000-721099 -->
        <alarmSpecification active="yes">
                <moduleId>NGEE.SM</moduleId>
                <errorCode>721000</errorCode>
                <severity>4</severity>
                <modelDescription>NGEE, Service is down</modelDescription>
                <activeDescription>NGEE, Service is down</activeDescription>
                <eventType>3</eventType>
                <probableCause>612</probableCause>
        </alarmSpecification>
        <alarmSpecification active="yes">
                <moduleId>NGEE.SM</moduleId>
                <errorCode>721001</errorCode>
                <severity>3</severity>
                <modelDescription>NGEE, Service cannot be
restarted</modelDescription>
                <activeDescription>NGEE, Service cannot be restarted
</activeDescription>
                <eventType>3</eventType>
                <probableCause>612</probableCause>
        </alarmSpecification>
        <!-- CM range is 721100-721199 -->
        <alarmSpecification active="yes">
```

```
                <moduleId>NGEE.CM</moduleId>
                <errorCode>721100</errorCode>
                <severity>3</severity> <!--critical-->
                <modelDescription>NGEE, Zookeeper is down or unreachable
</modelDescription>
                <activeDescription>NGEE, Zookeeper is down or unreachable
</activeDescription>
                <eventType>3</eventType> <!-- Quality of Service -->
                <probableCause>612</probableCause> <!-- Out of Service -->
        </alarmSpecification>

        <!-- PM range is 721200-721299 -->
</alarmDefinitions>
```

4.3 Alarm Notification Sequence

4.3.1 Application Invokes Execution Environment Java OaM API

```
NotificationLogger logger =
    NotificationLoggerFactory.getInstance(NotificationLogger4Ngee.class);
logger.postNotification(NgeeFmTest.class.getName(), 5, "1.1.1.100.5", "Error
message");

// Parameters of the above logger.postNotification()
// loggingClassName: name of the class to log the faults
//                   (usually the application class name)
// notificationID:   id that matches to the one defined in the ECIM model
// resourceID:       SNMP OID (e.g. 1.1.1.100.1, 1.1.1.100.2, etc.).
//                   uniquely identify your cluster, process, etc.
// message:          the fault message
```

4.3.2 Execution Environment OaM Notification2Alarm Raises Alarm Based on the ECIM FM Instance Configuration

The notification ID “5” in the API call in the previous section triggers an alarm based on the following ECIM Instance configuration:

```
       <object parentDn="Ngee=1,NgeeFm=1">
            <hasClass name="NgeeFmNotification"> <mimName>NgeeMom</mimName>
            </hasClass>
            <slot name="notificationId"> <value>5</value> </slot>
            <slot name="notificationDesc"> <value>NGEE, Service is down</value>
            </slot>
            <slot name="logLevel"> <value>ERROR</value> </slot>
            <slot name="moduleId"> <value>NGEE.SM</value> </slot>
            <slot name="notificationType"> <value>RAISE_ALARM</value> </slot>
            <slot name="errorCode"> <value>721000</value> </slot>
        </object>
```

4.3.3 Details of the Error Code in the ECIM FM Instance Configuration Defined in the Alarm Configuration File

The alarm configuration file contains the following details of error code 721000.

```
<alarmSpecification active="yes">
    <moduleId>NGEE.SM</moduleId>
    <errorCode>721000</errorCode>
    <severity>4</severity>
    <modelDescription>NEEE, Service is down</modelDescription>
    <activeDescription>NGEE, Service is down</activeDescription>
    <eventType>3</eventType>
    <probableCause>612</probableCause>
  </alarmSpecification>
```

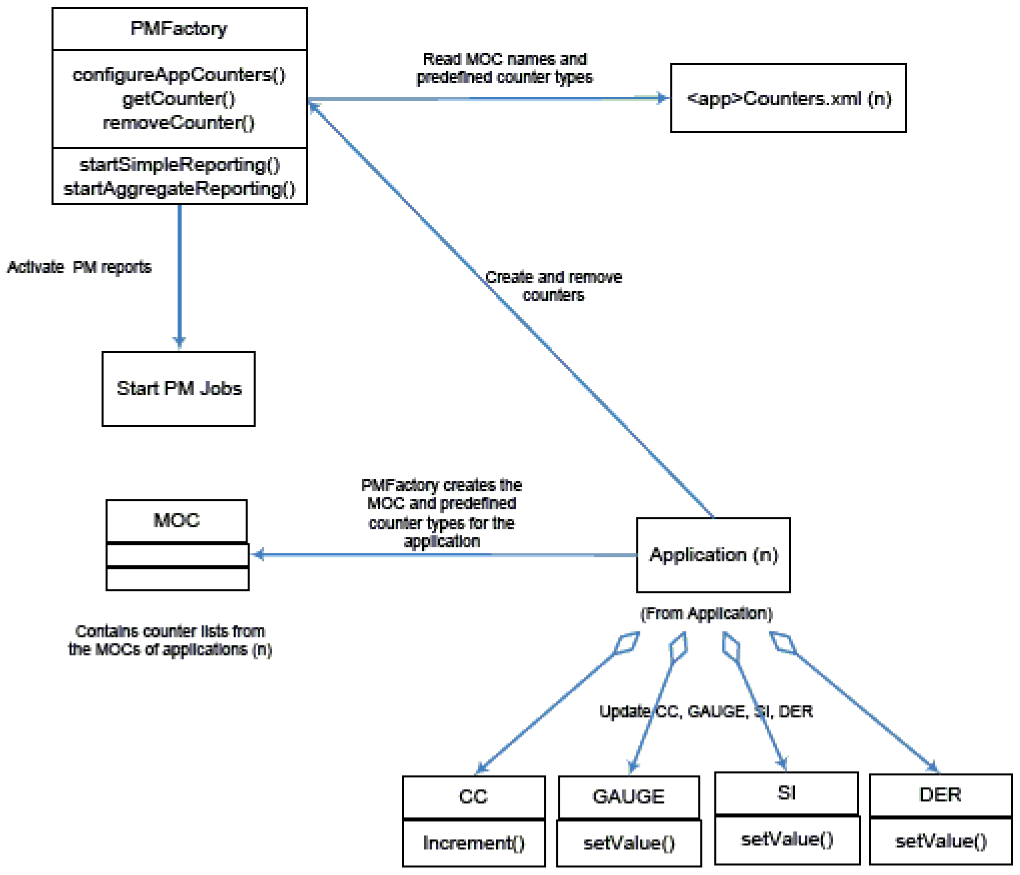
4.3.4 The ESA Command Line Tool Shows the Above Active Alarm

```
[root@SUF03 bin]# ./fmactivealarms
Active alarms:
!---------------------------------------------------------------
Module             : NGEE.SM
Error Code         : 721000
Resource Id        : 1.1.1.100.5
Timestamp          : Fri Sep 13 10:08:04 EDT 2013
Model Description  : NEEE, Service is down
Active Description : NGEE, Service is down
Event Type         : 3
Probable Cause     : 612
Severity           : major
Orig Source IP     : 192.168.10.21
---------------------------------------------------------------!
```

5 Performance Management

5.1 Overview of PM in Execution Environment

Ngee_boat provides a PMRuntime class for developers to use PM in Execution Environment; it is a simple wrapper of PMFactory. An overview of the implementation of PMFactory is shown in the following figure. Both PMRuntime and PMFactory can be used for implementation.


PMFactory
configureAppCounters()
getCounter()
removeCounter()
startSimpleReporting()
startAggregateReporting()
Read MOC names and predefined counter types
<app>Counters.xml (n)
Activate PM reports
Create and remove counters
Start PM Jobs
MOC
PMFactory creates the MOC and predefined counter types for the application
Application (n)
(From Application)
Contains counter lists from the MOCs of applications (n)
Update CC, GAUGE, SI, DER
CC
Increment()
GAUGE
setValue()
SI
setValue()
DER
setValue()

5.2 Performance Collection Methods

PM in Execution Environment supports the following performance collection methods:

- Cumulative Counter (CC)

  A cumulative counter is implemented with a 32 bit integer variable. The application updates the counter value using the increment() method.

- Gauge (Gauge)

  A gauge is implemented by either a 32 bit integer or 32 bit real variable. A gauge may have watermarks to store the minimum (MIN) and maximum (MAX) values measured in a certain granularity period. The application updates the gauge value using the setValue(value) method.

- Status Inspection (SI)

  Status Inspection (SI) is a data collection method where a value is sampled a number of times at fixed intervals within one granularity period. The sampling interval is fixed when the SI is instantiated, based upon the expected rate of change of the counter or gauge value.

  The sampled values are processed to calculate either the average, (MEAN), minimum (MIN), or maximum (MAX) for the granularity period. It is also possible to just report the value at the end of the granularity period (LATEST). SI measurements are reset at the beginning of the granularity period and only have a valid result at the end of the granularity period.

  The value sampled, as well as the calculated MEAN, MIN, MAX, or LATEST are implemented by either 32-bit integer or 32-bit real variables.

- Discrete Event Registration (DER)

  Discrete Event Registration (DER) is a data collection method where data, that is, a number related to a particular event, is captured. Each nth event is registered, where n can be 1 or larger. The value of n is fixed at the moment the DER is instantiated, based on the frequency of occurrence of the event being measured.

  The registered values are processed to calculate either the average (MEAN), minimum (MIN), or maximum (MAX) for the granularity period. It is also possible to just report the value at the end of the granularity period (LATEST). DER measurements are reset at the beginning of the granularity period and only have a valid result at the end of the granularity period.

  The value sampled, as well as the calculated MEAN, MIN, MAX, or LATEST are implemented by either 32-bit integer or 32-bit real variables.

5.3 Define Counters for Application

Each application provides an ***<App>Counters.xml*** file, where <*App*> is the name of the application. This file defines the Managed Object Class (MOC) names, and the above predefined counter types that are used or required by the application. As a result, the PM Factory creates the MOCs and predefined counter types for the application. Each MOC name defined must be unique within the cluster. If you instantiate a counter with undefined MOC name or counter name, PMException will be thrown.

A recommended MOC name must include both the application name and a logical subcomponent name used to group some counters. For consistency in the measurement output file where DNs are shown, the following syntax is recommended:

<*app-name*>=1,<*group-name*>

For example, “AVM=1,MetadataInterface” can be used for the counters on CwfMetadataInterface.

DN: Distinguished Name

Sample

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<appCounters>
      <moc>
                   <name>ExampleApp=1,ExampleMoc</name>                  MOC Name
follows the naming guidelines
                   <counter>
                                <name>ExampleCounter</name>
      Counter Name
                                <type>CC</type>
      Counter Type
                   </counter>

                   <counter>
                                <name>ExampleGauge</name>
                                <type>GAUGE</type>
                                <format>INTEGER</format>
                                <mode>ALL</mode>
                                <initValue>15</initValue>
                   </counter>

                   <counter>
                                <name>ExampleSi</name>
                                <type>SI</type>
                                <format>INTEGER</format>
                                <mode>ALL</mode>
                                <interval>3</interval>
                                <initValue>0</initValue>
                   </counter>
                   <counter>
                                <name>ExampleDer</name>
                                <type>DER</type>
                                <format>INTEGER</format>
                                <mode>ALL</mode>
```

```
                                        <interval>2</interval>
                                        <initValue>0</initValue>
                                </counter>
                        </moc>
                </appCounters>
```

5.4 PMRuntime Lifecycle

PMRuntime is instantiated by calling PMRuntime.init() when the application starts. It lives in the same VM as the application. After PMRuntime is started successfully, all counter definition files that end with **Counters.xml** in NEGG_DIR/conf/pm are loaded automatically.

Applications can also call PMFactory.configureAppCounters(InputStream) where InputStream is a stream of XML counter definitions to add new counters definition after PMRuntime is initialized.

```
InputStream is=new FileInputStream("./myCounters.xml");

PMFactory.configureAppCounters(is) ;

PMRuntime.start();
```

The application must stop PMRuntime before it exits by calling PMRuntime.stop();

Execution Environment scans /opt/ngee/latest/base/conf/pm directory for existing and newly created PM counter definitions inside PMRuntime lifecycle.

5.5 Create and Update Counter Instance

After successfully initializing PMRuntime and counter definitions being loaded, the application can create counter instance and update it.

```
PMRuntime.getCounter(String MocName, String CounterName,
String CounterInstanceName);
```

MocName must match the MOC defined by the application.

CounterName must match the counter name defined under MOC.

CounterInstanceName is the counter instance name being assigned.

The following code snippet shows how to create and update a counter instance.

```
CounterImpl cnt2=(CounterImpl)
PMRuntime.getCounter("Moc1", "Moc1Cntr1",
"CntrInstance1");

cnt.increment();
```

If a matched definition is not found, PMRuntime throws PMException.

5.6 Measurement Job

To collect data of counter instances, a measurement job needs to be created through CLI (Execution Environment EE Command Line Tool) by OSS admin.

Each job has a unique job ID. PMRuntime detects and processes incoming requests and sends data periodically to the master node.

The following shows an example of a measurement job result file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<measCollecFile>
     <fileHeader fileFormatVersion="3GPP_PM_32.435_XML_v7.2.0">
          <measCollec beginTime="2008-07-09T16:15:00.000+02:00"/>
     </fileHeader>
     <measData>
          <managedElement localDn="ALL=1,ExampleApp=1,ExampleMoc=Instance1"/>
          <measInfo>
               <job jobId="ExampleJob"/>
               <granPeriod endTime="2008-07-09T16:20:00.000+02:00"
duration="PT300S"/>
               <repPeriod duration="PT300S"/>
               <measType p="0">ExampleCounter</measType>
               <measValue measObjLdn="LDN">
                    <r p="0">4</r>
               </measValue>
          </measInfo>
     </measData>
     <measData>
          <managedElement localDn="traffic_instance_MTL-
3,ExampleApp=1,ExampleMoc=Instance1"/>
          <measInfo>
               <job jobId="ExampleJob"/>
               <granPeriod endTime="2008-07-09T16:20:00.000+02:00"
duration="PT300S"/>
               <repPeriod duration="PT300S"/>
               <measType p="0">ExampleCounter</measType>
               <measValue measObjLdn="LDN">
                    <r p="0">1</r>
               </measValue>
          </measInfo>
     </measData>
     <measData>
          <managedElement localDn="traffic_instance_MTL-
4,ExampleApp=1,ExampleMoc=Instance1"/>
          <measInfo>
               <job jobId="ExampleJob"/>
```

```
                        <granPeriod endTime="2008-07-09T16:20:00.000+02:00"
duration="PT300S"/>
                        <repPeriod duration="PT300S"/>
                        <measType p="0">ExampleCounter</measType><measValue
measObjLdn="LDN">
                             <r p="0">3</r>
                        </measValue>
                   </measInfo>
              </measData>
              <fileFooter>
                   <measCollec endTime="2008-07-09T16:20:00.000+02:00"/>
              </fileFooter>
         </measCollecFile>
```

The previous example shows the result of a job called ExampleJob, which has a granularity period of 5 minutes (300 seconds). The file reports the results recorded on June 09, 2008 at 16:15, for the MOC named ExampleApp=1,ExampleMoc. The results include the counter ExampleCounter with instance Instance1. The aggregated result is 4. This is the addition of 1 from the MTL-3 node, and 3 from the MTL-4 node.

5.7 AVM Implementation

5.7.1 AVM Counters Definition

AVMCounter.xml contains all predefined MOCs and counters of AVM. The following table shows how the MOC and counter names can be defined. This file is copied to $NGEE_INSTALL_DIR/conf/pm during installation.

Function	Type	MOC	Counter Name	CounterInstance Name	Interval
Received Msg	CC	AVM=1,MetadataInterface	ReceivedMessage	[path].ReceivedMessage	N/A
Invoke Count	CC	AVM=1,MetadataInterface	Invocation	[path].Invocation	N/A
Invoke Error	CC	AVM=1,MetadataInterface	InvokeError	[path].InvokeError	N/A
Processing Time	SI	AVM=1,MetadataInterface	ProcessTime	[path].ProcessTime	1
Enqueue Count	CC	AVM=1,Queue	EnqueueCount	qName.enqueue.counter	N/A
Deque Count	CC	AVM=1,Queue	DequeueCount	qName.dequeue.counter	N/A
Enqueue Error	CC	AVM=1,Queue	EnqueueError	qName.enque.error	N/A
Deque Error	CC	AVM=1,Queue	DequeueError	qName.dequeue.error	N/A
Received Msg	CC	AVM=1,MetadataInterfaceOperation	ReceivedMessage	[path].ReceivedMessage	N/A
Invoke Count	CC	AVM=1,MetadataInterfaceOperation	Invocation	[path].Invocation	N/A
Invoke Error	CC	AVM=1,MetadataInterfaceOperation	InvokeError	[path].InvokeError	N/A

Function	Type	MOC	Counter Name	CounterInstance Name	Interval
Processing Time	SI	AVM=1,MetadataInterfaceOperation	ProcessTime	[path].ProcessTime	1

5.7.2 CW Application Counters Definition

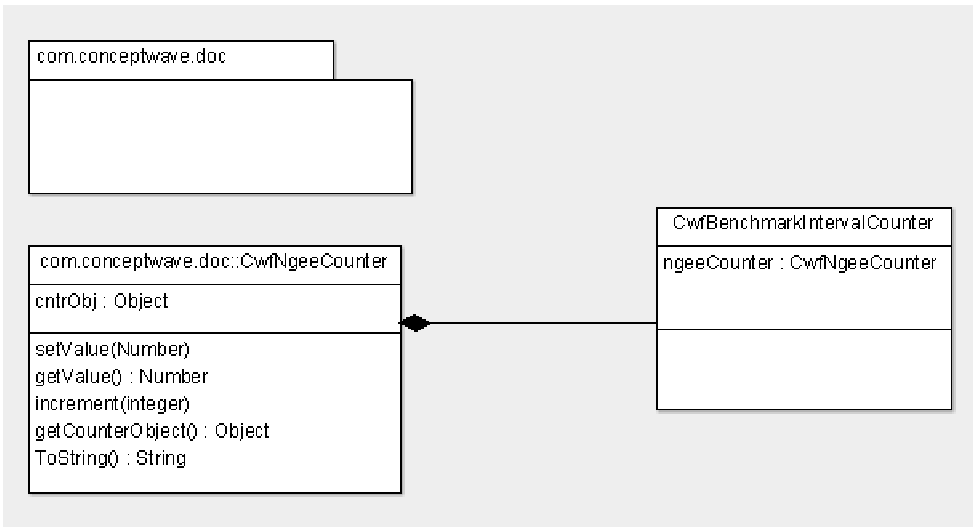
SUF installer of AVM, templates and application is responsible for copying counter definition files to the default PM configuration directory (opt/ngee/latest/base/conf).

Naming conventions are as follows:

- **avmCounters.xml**: contains all counter definitions for AVM.
- ***template*Counters.xml**: contains counter definitions for a template library. In the filename, *template* is replaced with the library name of a template, such as **sofCounters.xml**, **reportCounters.xml**, etc.
- ***app*Counters.xml**: contains application counter definitions. In the filename, *app* is replaced with the application name, such as **allStreamCounters.xml** or **rogersCounters.xml**.

When a counter definition file name follows the naming conventions and is located in the default PM configuration directory, the definitions contained in the file are merged by PMRuntime during its initialization.

5.7.3 Java Implementation


com.conceptwave.doc
com.conceptwave.doc::CwfNgeeCounter
cntrObj : Object
setValue(Number)
getValue() : Number
increment(integer)
getCounterObject() : Object
ToString() : String
CwfBenchmarkIntervalCounter
ngeeCounter : CwfNgeeCounter

- Initialize PMRuntime when AVM starts by calling PMRuntime.init().
- Stop PM Runtime when AVM stops.
- Add cwfNgeeCounter.

- Constructor: CwfNgeeCounter(String mocName,String counterName,String instName);
- Enhance CwfBenchmarkIntervalCounter to support Execution Environment, as follows:
  - Add ngeeCounter: CwfNgeeCounter, which maps to CwfBenchmarkIntervalCounter.totalEvents. The instance name is the counter name.
  - Add ngeeDuration: CwfNgeeCounter, which maps to CwfBenchmarkIntervalCounter.duration. The instance name is the counter name concatenating with .duration.
  - Add a new constructor to indicate if a counter in Execution Environment shall be created additionally.
  - public CwfBenchmarkIntervalCounter(CwfSystem sys, String domain, String name, boolean visibleInJMX, String Moc, String CounterDefName, String durationDefName);
  - Add getNgeeCounterValue() function for testing purposes.
  - Return ngeeCounter.getValue() if it's not null.
  - Add getNgeeDurationrValue() function for testing purposes.
  - Return ngeeDuriaton.getValue() if it's not null.
  - In increment(event,duration,cx) function, pass values to Execution Environment counters if they are created.
- Change places in CwfMetadataInterface and CwfQueue where instantiate CwfBenchmarkIntervalCounter, call the new constructor with hard coded MOC,CounterDefNames.
- Append parameters to CwpProcessManager and pass to: public CwfBenchmarkIntervalCounter addBenchmarkCounter(String counterName, boolean visibleInJMX, boolean dontCreateNew, String ngeeMOC, String ngeeCounterDef, String durationCounterDef)
- Remove Execution Environment counter in CwpProcessManager. removeBenchmarkCounter().
- If running in Execution Environment, benchmark counters on interface/queue shall always be created and functioning.

5.7.4 Import Execution Environment libraries

Include only the JARS needed for the build. At run time, $CLASSPATH must include Execution Environment library PATH. This option is needed to support running AVM in Execution Environment.

6 Trademarks

Ericsson, the Ericsson logo and the Globemark are trademarks of Ericsson.

Ericsson is a recognized leader in delivering communications capabilities that enhance the human experience, ignite and power global commerce, and secure and protect the world's most critical information. Serving both service provider and enterprise Customers, Ericsson delivers innovative technology solutions encompassing end-to-end broadband, Voice over IP, multimedia services and applications, and wireless broadband designed to help people solve the world's greatest challenges. Ericsson does business in more than 150 countries. For more information, visit Ericsson on the Web at www.Ericsson.com.

7 Disclaimer

This document may contain statements about a number of possible benefits that Ericsson believes may be achieved by working with Ericsson. These might include such things as improved productivity, benefits to end users or cost savings. Obviously, these can only be estimates. Gains might be qualitative and hard to assess or dependent on factors beyond Ericsson's control. Any proposed savings are speculative and may not reflect actual value saved. Statements about future market or industry developments are also speculative.

Statements regarding performance, functionality, or capacity are based on standard operating assumptions, do not constitute warranties as to fitness for a particular purpose, and are subject to change without notice.

This document contains Ericsson's proprietary, confidential information and may not be transmitted, reproduced, disclosed, or used otherwise in whole or in part without the expressed written authorization of Ericsson.