ESA Setup and Configuration

Ericsson SNMP Agent 16.0

SYSTEM ADMINISTRATION GUIDE

1/1543-CSH 109 532 Uen A

Contents

1 About This Document

1.1 Purpose

The purpose of this document is to describe the basic system administration tasks for the Ericsson SNMP Agent (ESA). The basic tasks means the mandatory tasks needed to get the ESA to run at all.

1.2 Target Group

The target group for this document is personnel responsible for administration of the ESA.

1.3 Prerequisites

It is assumed that the user of this document fulfils the following prerequisites.

- Has basic knowledge about SNMP.
- Is familiar with XML.
- Has system administrator authority to the server, in which the ESA is installed.
- If ESA cluster mode and SSL should be used, the reader is assumed to have knowledge about SSL configuration.

1.4 Typographic Conventions

The typographic conventions used in this document are described in Reference [1].

2 Configuration Work Process

2.1 Introduction

The ESA is built to be very flexible and to suit basically any environment. Flexibility come with a small cost though; The cost of handling configuration parameters.

Please note that an ESA is operational immediately after installation and does not really require changing any configuration parameter.

The following chapters describe the recommended and optional configuration procedures that are commonly performed at a newly installed ESA. Most of them are optional though.

See also Section 4 on page 15, which describes all the configuration parameters in more detail.

2.2 Mandatory Configuration

There are no mandatory configuration operations. The ESA is operational with default parameter settings immediately after installation.

2.3 Recommended Configuration

- **Trap Destination**

  The trap destination configuration specifies the target system, which is the OSS, that receives the SNMP notifications/traps sent by the ESA. One or several targets can be defined.

  Further details are found in Section 4.18 on page 67.

  Please note that the ESA actually can start with no trap destination at all, but having an ESA running without SNMP traps being sent is probably not very useful.

2.4 Optional Configuration

The optional configuration procedures are useful to know about when there are interfaces that must be changed, security is to be setup or other options and capabilities are to be defined according to specific system and/or customer needs.

The following configuration settings are optional:

- **AgentX and SNMP Proxy**

  The proxy is used for accessing subagents through the ESA Master Agent. Any existing SNMP based agent or AgentX based agent on the system may become a subagent to the ESA Master Agent. The subagents are connected to the Master Agent by either AgentX or by a SNMP Proxy. In both cases the proxy routes incoming SNMP operations to the subagent that represent the OID present in the operation.

  See Section 4.10 on page 48 for AgentX settings and Section 4.15 on page 61 for proxy settings.

- **Alarm Controllers**

  The options Alarm Clear Control (ACC) and Alarm Flooding Control (AFC) makes ESA suppress faulty alarm sequences. ACC and AFC are active by default.

  See Section 4.3 on page 16.

- **Cluster Mode**

  The ESA can operate in cluster mode, if needed. By default the cluster mode is inactive.

  See Section 4.4 on page 20.

- **Community Strings**

  The community string settings for the SNMP v1 and v2c traffic are set to default values. The community strings in the OSS and in the ESA must be identical.

  See Section 4.5 on page 37.

- **Directory Settings**

  The ESA provides the possibility to change the location of the configuration directories for the FM Agent and the PM Agent as well as the output directories for the PM Agent.

  See Section 4.6 on page 40.

- **FM MIB Preferences**

  The ESA Fault Management MIBs provides a couple of parameters related to log size of sent messages.

  See Section 4.7 on page 43.

- **Information Agent**

The ESA Information Agent (IA), which is included in the Master Agent, can be setup to publish system information on the SNMP interface. The system information is configurable in configuration files. The Information Agent is also publishing ESA information, but that data is provided by the ESA itself and is not configurable.

See Section 4.8 on page 45.

- **Interfaces**

  When the ESA is deployed on a server with multiple IP addresses you can bind the interfaces to a specific IP address. This is done in order to stop users from using the ESA services from not authorized IP addresses. The port settings are set to default values. Changing the port values are only needed in case a port collision occurs when other applications use the same ports, or if own values are desired, for example due to security reasons.

  See Section 4.9 on page 46.

- **Java Virtual Machine**

  There are a few JVM options that can be set for running the ESA.

  See Section 4.13 on page 55.

- **PM Agent Preferences**

  The PM Agent provides a few configuration parameters related to the creation of 3GPP XML files.

  See Section 4.14 on page 58.

- **User-based Security Model (USM)**

  The ESA supports setting up USM user configurations.

  See Section 4.19 on page 72.

- **View-based Access Control Model (VACM)**

  The ESA supports setting up VACM configurations.

  See Section 4.20 on page 75.

- **SNMPv3**

  If SNMPv3 is to be used, the ESA must be configured to work in SNMPv3 mode.

  See Section 4.16 on page 64.

- **SNMPv3 and VIP**

If SNMPv3 and VIP is to be used, the ESA Master Agent cluster mode needs to be enabled.

See Section 4.4 on page 20.

- **Log Files**

  The ESA log files can be customized by modifying log file handling configurations.

  See Section 5.2 on page 81.

2.5 Feature Configurations

The configurations in the previous chapters are related to the execution of the ESA itself. To get the most out of the ESA a number of configuration files should be setup for the FM and PM features provided by the ESA.

For FM the following configurations could be setup. Also, see Reference [2].

- **Alarm Definition**

  This part defines all the alarms that can be sent by the ESA FM Agent.

  Please note that the ESA FM Agent can start with no alarm definitions at all. This would be the case if only the PM features are used in the ESA and the FM features are made inactive.

- **Alarm Translation**

  This part defines all the alarm translations in the FM Agent for handling incoming traps and notifications from subagents.

  Please note that the ESA FM Agent can start with no alarm translations at all.

For PM the following configurations could be setup. Also, see Reference [3].

- **Counter Definition**

  This part defines all the counters that can be managed by the ESA PM Agent.

  Please note that the ESA PM Agent can start with no counter definitions at all. This would be the case if only the FM features are used in the ESA and the PM features are made inactive.

- **Counter Jobs**

  This part defines all the counter jobs that shall be managed by the ESA PM Agent.

Please note that the ESA PM Agent can start with no counter jobs at all.

- **Counter Thresholds**

  This part defines all the counter thresholds that shall be managed by the ESA PM Agent.

  Please note that the ESA PM Agent can start with no counter thresholds at all.

3 Operation

3.1 Introduction

The following chapters describes the ESA services and how to operate them.

3.2 Services

There are three ESA services running.

- ESA Master Agent service

  Service name: `esama`
- ESA FM Agent service

  Service name: `esafma`
- ESA PM Agent service

  Service name: `esapma`

The name of the services are the same on all platforms.

3.3 Operations

3.3.1 Overview

All ESA services support the following operations.

- Start
- Stop
- Restart
- Status

3.3.2 Linux

The ESA Linux services are operated by using the following instructions.

`Command format:`

`service <service name> <operation>`

Attribute <operation> allows the following values:

start	Start the specified service.
stop	Stop the specified service.
restart	Restart the specified service.
status	View the status of the specified service.

Command example:

```
# service esama start
# service esafma stop
# service esapma restart
# service esama status
```

Please note that the output from the command service is different for different Linux variants.

3.3.3 Solaris

The ESA Solaris services are operated by using the following instructions.

Command format:

svcadm *<service name> <operation>*

Attribute <operation> allows the following values:

enable	Start the specified service.
disable	Stop the specified service.
restart	Restart the specified service.

Command example:

```
# svcadm enable esama
# svcadm disable esafma
# svcadm restart esapma
```

To check the service status the following command is used.

Command format:

svcs *<option> <service name>*

Common values for <option> are the following:

(blank)	Option left blank. The service status is shown.
-l	Displays all available information about the service and service instances.
-p	Lists processes associated with each service instance.

See man pages for command svcs to use other service viewing options.

Command examples:

```
# svcs esama
# svcs -l esama
# svcs -p esama
```

3.3.4 Windows

The ESA Windows services are operated by using the following instructions.

☐ Either the Service Manager in the Control Panel is used, or command line such as the following.

Command format:

net *<operation> <service name>*

Attribute <operation> allows the following values:

start	Start the specified service.
stop	Stop the specified service.
restart	Restart the specified service.
status	View the status of the specified service.

Command examples:

```
C:\> net start esama
C:\> net stop esafma
C:\> net restart esapma
C:\> net status esama
```

3.4 Startup/Shutdown Sequence

From ESA point of view there is no required startup or shutdown sequence of the ESA services. The services run independently of each other meaning if one or several services are not up and running, the services will handle the missing relation and, most important, they will connect when the services are back in operation.

However, to optimize the startup and shutdown sequences and to also avoid warning and error indications in the logs, the following sequences are *recommended*.

Recommended Startup Sequence:

1 esama

2 esafma

Depends on the esama for AgentX connection and for alarm synchronization from the OSS.

3 esapma

Depends on the esama for AgentX connection and on esafma for counter threshold processing (triggering alarms).

4 ssmagent

Depends on the esama for handling OSS requests to the SSM and also esafma for sending traps to the OSS.

Recommended Shutdown Sequence:

1 ssmagent

2 esapma

3 esafma

4 esama

3.5 Configuration

For Solaris the service configuration is found in the following files.

- ESA Master Agent service configuration

  `{esa basedir}/bin/esama.xml`

- ESA FM Agent service

  `{esa basedir}/bin/esafma.xml`

- ESA PM Agent service

  `{esa basedir}/bin/esapma.xml`

3.6 SNMP Operation

During the initialization of the ESA services, there is a coldstart trap sent to all configured trap destinations. The coldstart trap is a standard trap with trap OID .1.3.6.1.6.3.1.1.5.1 and is described in detail in RFC 1907.

The coldstart trap from the ESA is commonly used in the OSS. It is used as a detection mechanism for node restart and/or agent restart. The coldstart trap indicates an agent restart, but can in fact be an agent restart due to a node restart. From the coldstart trap it is not possible to detect if only the agent or the node restarted. Nevertheless, at any restart type the OSS should use the coldstart trap as a trigger to initiate an alarm synchronization procedure to synchronize the alarms between the manager (OSS) and the agent (ESA) when the node and agent are back in operation.

4 Configuration

4.1 Introduction

The ESA comes with quite many configuration parameters. The many configuration parameters of course makes the use of ESA very flexible, but high flexibility also come with higher complexity.

The following sub sections describe how the ESA may be configured when it comes to options, behavior and security. The chapters are in alphabetical order and may be performed independently of each other. If there is a relation between configuration operations it is clearly indicated in the instructions.

4.2 Configuration Files

The following configuration files come with the ESA. Please note that each configuration file (except cluster.conf) come with an associated XML Schema, which shall/should be used for configuration format verification. Both configuration files and XML Schemas are found in *{esa basedir}*/conf.

- Main

  This file contains the vital parameters that makes the ESA running at all.

  File name: `mainCfg.xml`

  XML Schema: `mainCfg.xsd`

- Trap Destination

  This file contains the trap destinations.

  File name: `trapDestCfg.xml`

  XML Schema: `trapDestCfg.xsd`

- Community

  This file contains the community string settings.

  File name: `communityCfg.xml`

  XML Schema: `communityCfg.xsd`

- Proxy

  This file contains the proxy settings.

File name: `proxyCfg.xml`

XML Schema: `proxyCfg.xsd`

- USM

  This file contains the USM settings.

  File name: `usmCfg.xml`

  XML Schema: `usmCfg.xsd`

- VACM

  This file contains the VACM settings.

  File name: `vacmCfg.xml`

  XML Schema: `vacmCfg.xsd`

- System Information

  This file contains the system information data.

  File name: `infoSystem.xml`

  XML Schema: `infoSystem.xsd`

- Cluster Mode

  This file contains the communication settings for the cluster mode.

  File name: `cluster.conf`

- Logging

  Logging and its configuration is described in a separate chapter.

  See Section 5.2 on page 81.

4.3 Alarm Controllers

4.3.1 Introduction

The alarm controllers are available in the ESA to control the alarm flow to the OSS receiving the alarms to the ESA.

The following alarm controllers are available:

- Alarm Clear Control (ACC)

- Alarm Flooding Control (AFC)

4.3.2 Function Description

4.3.2.1 Alarm Clear Control

The Alarm Clear Control (ACC) option makes it possible to stop *alarm clear* SNMP messages if no corresponding *alarm raise* SNMP message have been sent before.

The option is useful to stop faulty alarm sequences being sent to the receiving OSS. The OSS does not really need clearing of an alarm that is not active.

It is possible to configure whether the option is active or inactive. The option is by default active.

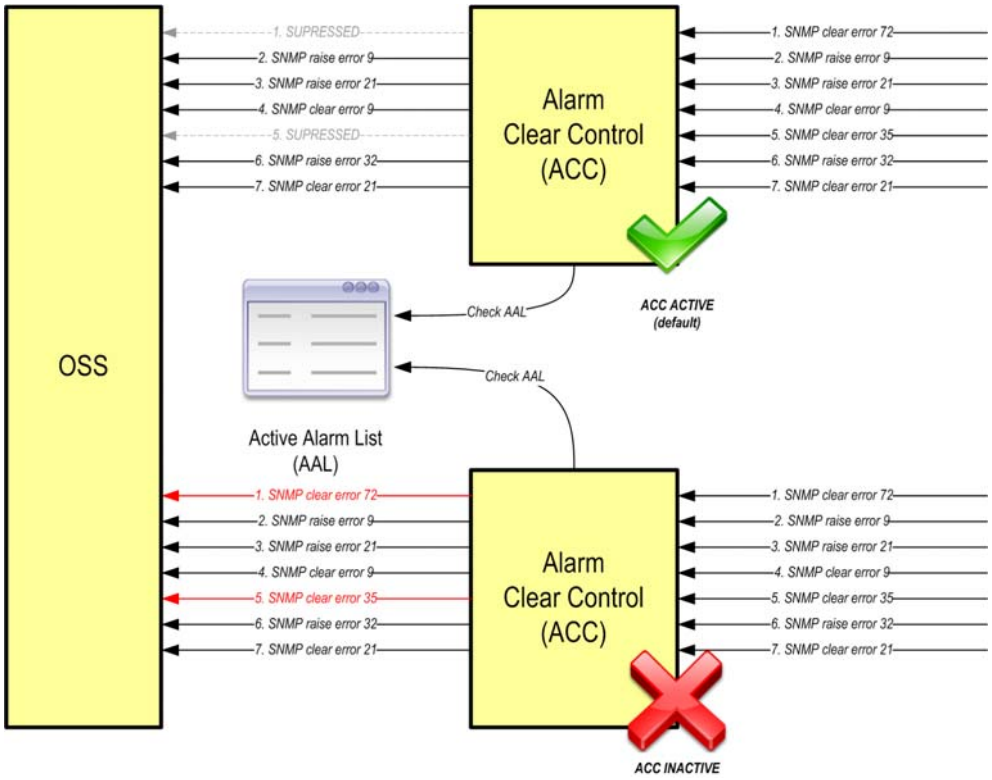

Figure 1 The Alarm Clear Control (ACC) function suppresses any alarm clear that does not have a corresponding alarm raise.

The figure above shows how the *alarm clear* without a corresponding *alarm raise* is suppressed by the ACC function in the ESA when the option is active. When inactive the *alarm clear* is not suppressed. When active, the FM Agent uses the Active Alarm List (AAL) to check if there is a matching *alarm raise* in the AAL. If there is not, the *alarm clear* is suppressed.

The alarm clear is matched with an alarm raise with regards to Module Identity, Error Code, Resource Identity and Originating Source IP.

4.3.2.2 Alarm Flooding Control

The Alarm Flooding Control (AFC) option makes it possible to avoid raising an identical and already active alarm being reported over and over again.

The option is useful to stop repeated alarms flooding the receiving OSS.

It is possible to configure whether the option is active or inactive. The option is by default active.

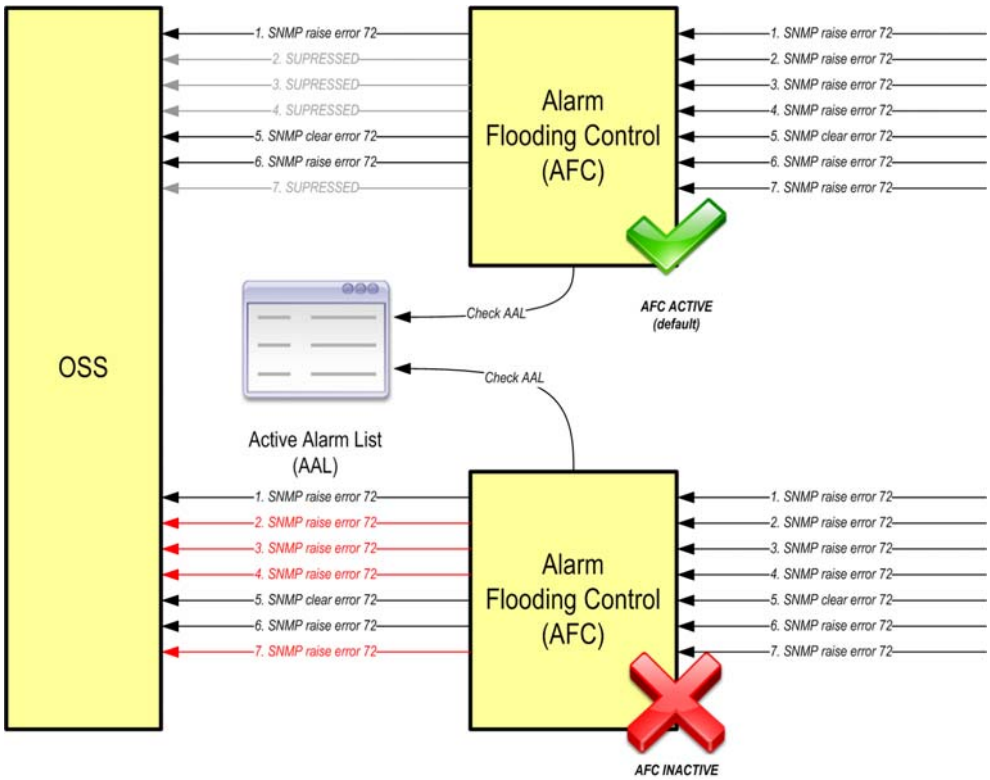

Figure 2 The Alarm Flooding Control (AFC) function suppresses repeated alarms.

The figure above shows how repeated alarms are suppressed by the AFC function in the ESA when the option is active. When inactive the repeated alarms are not stopped. When active, the FM Agent uses the Active Alarm List (AAL) to check if the alarm to be sent matches any alarm already in the AAL. If a matching alarm is found, the alarm that were supposed to be sent is suppressed.

The alarms are compared with regards to Module Identity, Error Code, Resource Identity and Originating Source IP.

4.3.3 Configuration File

The alarm controller parameters are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.3.4 Configuration Format

The alarm controller configurations are defined using the following format.

```
Configuration format:

<fm>
  <controllers>
    <acc>data</acc>
    <afc>data</afc>
  </controllers>
</fm>
```

The parameters contain the following data:

- **acc**

on	The ACC feature is active.
off	The ACC feature is inactive.

- **afc**

on	The AFC feature is active.
off	The AFC feature is inactive.

Examples are found in Section 4.3.6 on page 19.

4.3.5 Configuration Activation

The alarm controller configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.3.6 Examples

The following is a configuration example for the ESA alarm controllers.

```
<fm>
  <controllers>
    <acc>on</acc>
    <afc>on</afc>
  </controllers>
</fm>
```

4.4 Cluster Mode

4.4.1 Introduction

The Cluster Mode is a capability in the ESA that allows multiple ESAs to work together as one single unit. From an OSS view the cluster running multiple nodes will be seen as a single node.

With the cluster mode there are additional capabilities introduced, such as High Availability, Alarm Redundancy and Alarm Persistency.

4.4.2 Function Description

The Cluster Mode in the ESA means that the ESA agents in a cluster (two or more nodes) will work together as one single unit representing the cluster as one system. This is also known as "One System View".

One or several nodes will be defined as masters. The rest are simply non-masters of the cluster. The Alarm data will be synchronized over all ESA masters in the cluster.

All members in a cluster can trigger new alarms, but only the masters will process the alarms. What is also given with the Cluster Mode capability is support for High Availability, Alarm Redundancy and Alarm Persistency.

- **High Availability**

  The *High Availability* capability provides multiple ESAs to cooperate when it comes to handling the O&M operations. A switch over occurs from the active ESA master to one of the other defined, and currently standby, ESA master(s) in case the active master is lost.

- **Alarm Redundancy**

  The *Alarm Redundancy* capability makes the ESAs in the cluster holding the alarm data redundant by replicating the data over the masters.

- **Alarm Persistency**

  The *Alarm Persistency* capability makes the ESA holding the alarms persistent (active) after an ESA or node restart as long as at least one ESA master in the cluster is running. Please note that this is feasible only in Cluster Mode.

  The capability to have alarms persistent can be configured. By default alarm persistency is inactive.

  When alarm persistency is set *inactive*, a lost ESA or a lost node is detected by the active master in the cluster. The active master will then clear all active alarms for the node. When the lost ESA or lost node is

restarted, there are no alarms active for the node. Please note that this in fact is a normal behavior. The ESA cannot know if the ESA or the node has been down for five seconds or five hours. Having alarms persistent after a downtime of minutes or hours is not really reasonable.

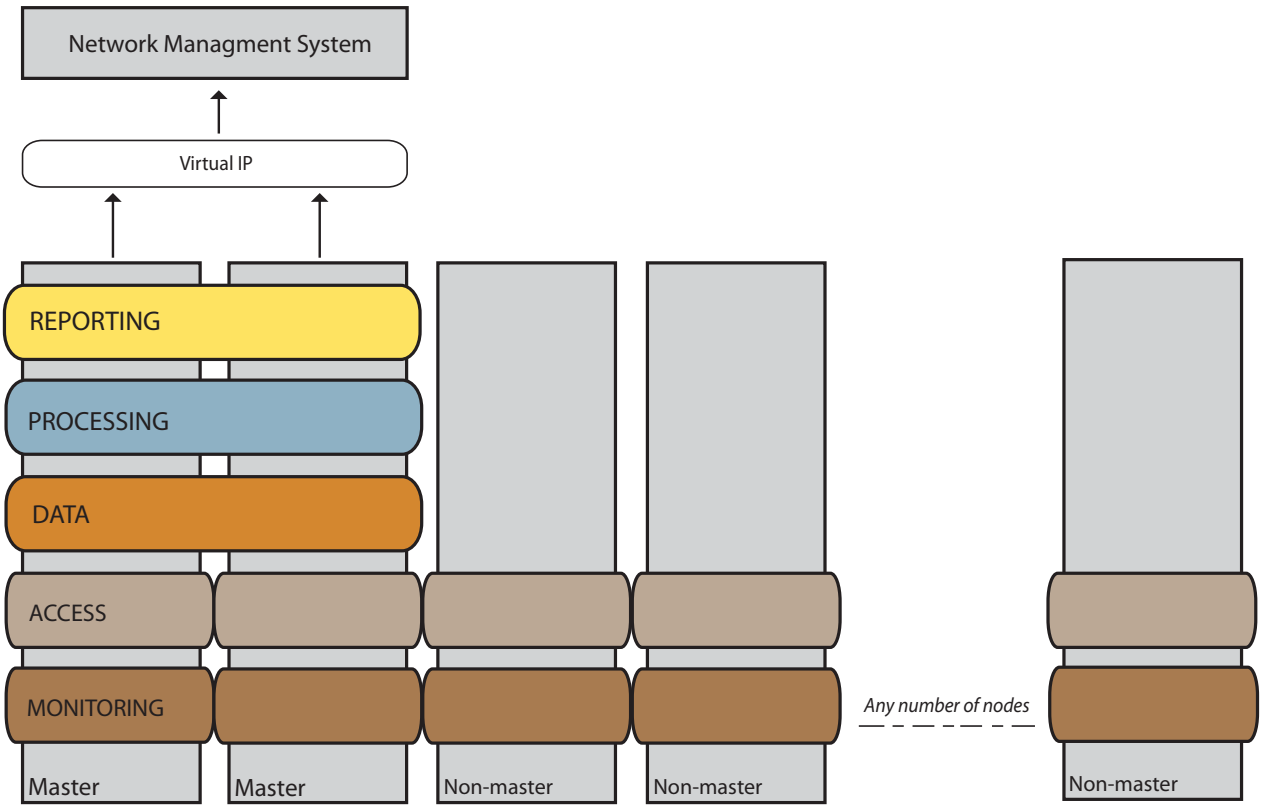

Figure 3 Overview of the ESA layers when operating in Cluster Mode.

The layers in the figure above have the following responsibilities.

- **Monitoring layer**

  This layer is about monitoring components, such as different monitoring agents. Example agents are SSM (optional component for the ESA), hardware monitoring agents and database monitoring agents. The monitoring is done locally on each node and the alarm/event reporting is done to the local access layer.

- **Access layer**

  This layer is local to each node, which means the application or component triggering alarms in the ESA is doing that to the ESA that is running locally on the node where the application or component is running.

- **Data layer**

  This layer is synchronized over the masters in the cluster. Alarms that are triggered in the access layer ends up in an alarm queue in the data

layer, which means the queue is stored only for the masters in the cluster. The data layer also holds the processed data from the operations in the processing layer.

The communication links in this layer are built using the Akka technology. For more information, see Reference [5].

- **Processing layer**

  This layer is operational only for the masters in the cluster. The processing capabilities are inactive on the other nodes (non-master nodes).

  There is always only one master being active. The other ESAs defined as masters are inactive. If the active master goes down another master takes the active role.

  The active master in the processing layer is the one working on the alarm queue in the data layer. The alarms are read one by one according to FIFO principle and processed according to the ESA configuration settings and current alarms status in the ESA. The processed status is stored in the data layer. Again, the processed data is synchronized over all masters in the cluster, which means the current alarm status for the entire system is the same no matter from which master it is checked.

- **Reporting layer**

  This layer is tightly connected to the processing layer, but still drawn as a separate layer. The master processing the alarms from the alarm queue is the one sending the SNMP alarms to the NMS.

  Please note that this setup *may* require the use of a VIP in the cluster setup. The alarms sent to the NMS *should* come from the same IP address no matter which node sent the alarms. If not, the NMS *may* not understand that the alarms from different masters are actually representing the same cluster. Some NMSes do however have the capability to combine the alarm flow from several IPs and handle it as one "single node".

The following is a simple example of an alarm being triggered in a non-master, but processed by one of the masters. All data is at all times synchronized over the masters in the cluster, both the alarms being triggered and the processed and active alarm data.

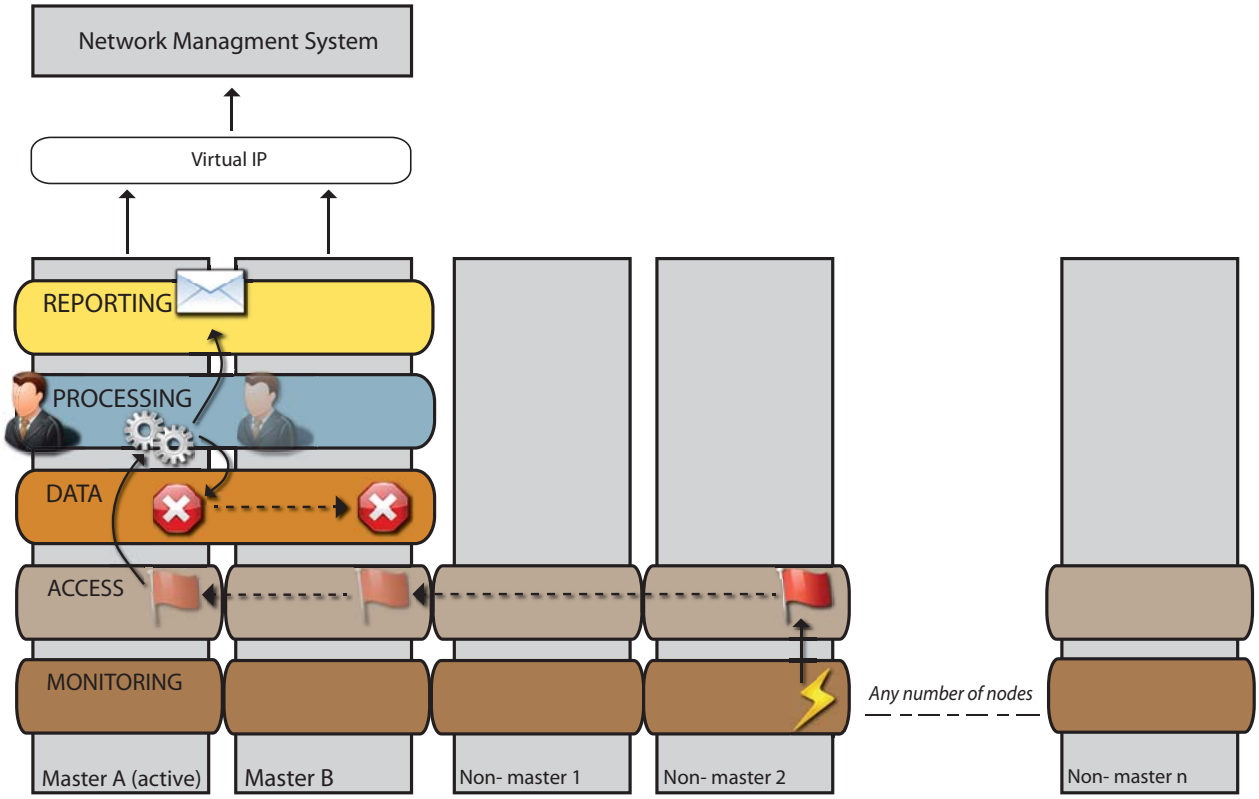

Figure 4 An example alarm flow in cluster mode

1. An error event is detected in a non-master.
2. The error is reported to the local ESA.
3. The error is placed in the alarm queue.
4. The master A is active and initiates the processing of the alarm queue.
5. The alarm is processed and stored in the Active Alarm List (AAL), which is replicated over the masters in the cluster.
6. The alarm is finally sent to the NMS.

Please note that if there is no master at all running in the cluster, all alarms being triggered are queued in the alarm queue in the local ESA. When a master is back in operation, the master will work on the alarm queue by processing the alarms one by one.

4.4.2.1 Active alarms in the cluster

The FM MIB data and the Active Alarm List (AAL) are only kept on the master nodes. This means that SNMP requests for alarms must be directed to one of the ESA master nodes. The command `$fmactivealarms` must be executed on a master node in the cluster. It will then fetch the AAL from the active

master. Executing the command `$fmactivealarms` on a non master node in the cluster will return an empty list as result, which is the expected behavior.

4.4.2.2 Master Agent / FM Agent cluster

The ESA Cluster Mode is actually running in two different clusters. The FM Agent cluster is handling the FM Agent functionality such as Alarm data. The Master Agent Cluster only handles VIP functionality. In order to use a VIP together with SNMPv3, the Master Agent Cluster needs to be active. For cluster solutions that operates a VIP together with SNMPv1 or SNMPv2c, or not running a VIP at all, Master Agent cluster should not be active.

4.4.3 Configuration Files

The cluster parameters are defined in the configuration files `mainCfg.xml` and `cluster.conf` in directory *{esa basedir}*`/conf`.

4.4.4 Configuration Format

4.4.4.1 Feature Configuration, mainCfg.xml

The main configurations are defined using the following format.

```
Configuration format:

<cluster active="data" master="data">
  <ma active="data">
  <fm active="data">
    <alarmPersistency>data</alarmPersistency>
  </fm>
</cluster>
```

The parameters contain the following options:

- **`cluster/active`**

  This attribute indicates if the cluster mode in the ESA is active or inactive. If set to inactive, the ESA works in single mode (also known as stand-alone mode).

`yes`	The cluster mode is active.
`no`	The cluster mode is inactive.

- **`cluster/master`**

  This attribute indicates if the ESA is a master or non-master.

`yes`	ESA is one of the masters in the cluster.
`no`	ESA is not one of the masters in the cluster.

- **`cluster/ma/active`**

  This attribute indicates if the Master Agent cluster feature is active. If set inactive, the Master Agent cluster works in single mode only.

`yes`	The Master Agent operate in cluster mode when the **`cluster/active`** is set to `"yes"`.
`no`	The Master Agent operate in single mode even though the **`cluster/active`** is set to `"yes"`.

- **`cluster/fm/active`**

  This attribute indicates if the Fault Management cluster feature is active. If set inactive, the Fault Management works in single mode only.

`yes`	The FM features operate in cluster mode when the **`cluster/active`** is set to `"yes"`.
`no`	The FM features operate in single mode even though the **`cluster/active`** is set to `"yes"`.

- **`cluster/fm/alarmPersistency`**

  This element indicates if the ESA shall hold the alarms persistent in an ESA restart. If set to `off`, the alarms are cleared at each ESA restart.

`on`	Alarm persistency is active.
`off`	Alarm persistency is inactive.

Configuration example:

```
<cluster active="yes" master="yes">
  <ma active="no">
  <fm active="yes">
    <alarmPersistency>off</alarmPersistency>
  </fm>
</cluster>
```

4.4.4.2 Feature Configuration, cluster.conf

Since the Master Agent and the FM Agent operates in different clusters, the `cluster.conf` configuration file contains parameters for both the Master Agent cluster and the FM Agent cluster. The configuration between the two are very similar, with the two exceptions: the **port** parameter and the **cluster name**. It is important that the two clusters are using different ports and cluster

names. This chapter concerns configuration formats for both the FM Agent cluster and the Master Agent Cluster.

4.4.4.2.1 Feature Configuration, cluster.conf, TCP

The most important Akka configurations are defined using the following format. For more information about configuration of Akka, see Reference [5].

```
Configuration format:

Agent{
  cluster {
     seed-nodes = [
       "akka.tcp://cluster_name@seed_ip_address:seed_port",
        ...]
  }
  ...
  remote {
    enabled-transports = ["akka.remote.netty.tcp"]
    netty {
      tcp {
        hostname = "ip_address"
        port = port
      }
    }
  }
}
```

The parameters contain the following data:

- **`Agent`**

  `Agent` in the example above is applicable to both Master Agent and FM Agent. Both these agents use the same configuration format.

- **`hostname`**

  This parameter indicates which hostname or IP address Akka will use.

  Note: The hostname can either be an IP address or a hostname

- **`port`**

  This parameter indicates which port Akka will use.

  Note: The Master Agent and FM Agent clusters can ***not*** use the same port

- **`seed-nodes`**

  This parameter is a list of all nodes in the ESA cluster.

`cluster_name`	Name of the ESA cluster system.
`seed_ip_address`	IP address of the cluster member.
`seed_port`	Port used for Akka communication.

Note: Seed nodes working in the same cluster **must** have the same cluster name. However, it is important to **not** use the same cluster name for two different clusters. The Master Agent cluster and FM Agent cluster **must not** have the same cluster names.

When starting a new cluster, it is important that each node has its **own** IP address or hostname as the **first** seed node in the seed-nodes lists.

Note: If a hostname is used in the `hostname` tag, please note that a hostname **must** also be used in the `seed_ip_address`, you **cannot** have a hostname in one tag and an IP address in the other.

The following settings of ***tcp*** indicates that TCP is used as transport protocol:

enabled-transports = ["akka.remote.netty.***tcp***"]

netty { ***tcp {..}***

akka.***tcp***://cluster_name@seed_ip_address:seed_port

Configuration example:

```
FmAgent{
  ...
  cluster {
    seed-nodes = [
      "akka.tcp://FMCluster@10.1.1.1:2551",
      "akka.tcp://FMCluster@10.1.1.2:2551",
      "akka.tcp://FMCluster@10.1.1.3:2551"]
  }
  remote {
    enabled-transports = ["akka.remote.netty.tcp"]
    netty {
      tcp {
        hostname = "10.1.1.1"
        port = 2551
      }
    }
  }
...
}

MasterAgent{
  ...
  cluster {
    seed-nodes = [
      "akka.tcp://MACluster@10.1.1.1:2552",
      "akka.tcp://MACluster@10.1.1.2:2552",
      "akka.tcp://MACluster@10.1.1.3:2552"]
  }
  remote {
    enabled-transports = ["akka.remote.netty.tcp"]
    netty {
      tcp {
        hostname = "10.1.1.1"
        port = 2552
      }
    }
  }
...
}
```

Note: As the example above shows, the Master Agent and FM Agent configuration will normally be very similar. Note that the only things that separates the two, are the ports and the cluster names.

4.4.4.2.2 Feature Configuration, cluster.conf, SSL

The most important Akka configurations are defined using the following format. For more information about configuration of Akka, see Reference [5]. The following configuration is applicable to both FM Agent and Master Agent cluster configuration.

```
Configuration format:

Agent{
  ...
  cluster {
      seed-nodes = [
        "akka.ssl.tcp://cluster_name@
             seed_ip_address:seed_port"
      ]
      ...
  }
  remote {
    enabled-transports = ["akka.remote.netty.ssl"]

    netty {
       ssl {
         enable-ssl = true
         hostname = "ip_address"
         port = port

         security {
           enabled-algorithms = ["enabled-algorithms"]
           key-password = "key_password"
           key-store = "keystore"
           key-store-password = "keystore_password"
           protocol = "protocol"
           random-number-generator = "random-number-gen"
           trust-store = "truststore"
           trust-store-password = "truststore_password"
         }
      }
    ...
    }
  }
}
```

The parameters contain the following data:

- **Agent**

  Agent in the example above is applicable to both Master Agent and FM Agent. Both these agents use the same configuration format.

- **hostname**

This parameter indicates which hostname or IP address Akka will use.

Note: The hostname can either be an IP address or a hostname

- **`port`**

  This parameter indicates which port Akka will use.

  Note: The Master Agent and FM Agent clusters can ***not*** use the same port

- **`security`**

`key-store`	Path to key store.
`key-store-password`	Password for key store.
`key-password`	Password for key.
`trust-store`	Path to trust store.
`trust-store-password`	Password for trust store.
`protocol`	Encryption Protocol. For available encryption protocols see Reference [5].
`random-number-gen`	Random number generator. For options see Reference [5].
`enabled-algorithms`	Encryption algorithms. For available algorithms see Reference [6].

  Note: The reader is assumed to have knowledge about SSL configuration.

- **`seed-nodes`**

  This parameter is a list of all master nodes in the ESA cluster.

`cluster_name`	Name of the ESA cluster system.
`seed_ip_address`	IP address or hostname of ESA.
`seed_port`	Port used for Akka communication.

  Note: Seed nodes working in the same cluster **must** have the same cluster name. However, it is important to **not** use the same cluster name for two different clusters. The Master Agent cluster and FM Agent cluster **must not** have the same cluster names.

  When starting a new cluster, it is important that each node has its **own** IP address or hostname as the **first** seed node in the seed-nodes lists.

Note: If a hostname is used in the `hostname` tag, please note that a hostname **must** also be used in the `seed_ip_address`, you **cannot** have a hostname in one tag and an IP address in the other.

The following settings of ***ssl*** indicates that SSL is used:

enabled-transports = ["akka.remote.netty.***ssl***"]

netty {***ssl {}***

enable.***ssl*** = true

akka.***ssl***.tcp://cluster_name@seed_ip_address:seed_port

Configuration example FmAgent:

```
FmAgent{
 ...
 cluster {
     seed-nodes = [
       "akka.ssl.tcp://FMCluster@10.1.1.1:2551",
       "akka.ssl.tcp://FMCluster@10.1.1.2:2551"]
 ...
 remote {
    enabled-transports = [
       "akka.remote.netty.ssl"]

    netty {
       ssl {
          enable-ssl = true
          hostname = "10.1.1.1"
          port = 2551

          security {
             enabled-algorithms =
                ["TLS_RSA_WITH_AES_128_CBC_SHA"]
             key-password = "changethis"
             key-store = "/opt/esa/conf/keystore"
             key-store-password = "changethis"
             protocol = "TLSv1"
             random-number-generator = ""
             trust-store = "/opt/esa/conf/truststore"
             trust-store-password = "changethis"
          }
       }
    }
 }
...
}
```

Configuration example MasterAgent:

```
MasterAgent{
 ...
 cluster {
     seed-nodes = [
       "akka.ssl.tcp://MACluster@10.1.1.1:2552",
       "akka.ssl.tcp://MACluster@10.1.1.2:2552"]
 ...
 remote {
    enabled-transports = [
       "akka.remote.netty.ssl"]

    netty {
       ssl {
          enable-ssl = true
          hostname = "10.1.1.1"
          port = 2552

          security {
             enabled-algorithms =
                ["TLS_RSA_WITH_AES_128_CBC_SHA"]
             key-password = "changethis"
             key-store = "/opt/esa/conf/keystore"
             key-store-password = "changethis"
             protocol = "TLSv1"
             random-number-generator = ""
             trust-store = "/opt/esa/conf/truststore"
             trust-store-password = "changethis"
          }
       }
    }
 }
...
}
```

4.4.5 Configuration Activation

The cluster mode configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

After configuration and ESA start it could be useful to run the status check command to verify that the nodes in the cluster are up and running. The command will show which nodes are successfully communicating in the ESA cluster. See Section 4.4.7 on page 33.

4.4.6 Examples

The Akka configuration is in detail described in Reference [5].
Examples of cluster configuration can be found in the configuration files `example_cluster.conf` and `example_cluster_SSL.conf` in directory *{esa basedir}*`/conf/examples`.

4.4.7 Command: Cluster Status

4.4.7.1 Description

This command is used for viewing the current cluster status. It will present all the members and the member identities. Also, the output shows which members are masters and which master is the active one.

4.4.7.2 Command Format

The command format:

```
$esaclusterstatus <input>
```

<input> This input is optional. Leaving the `"$esaclusterstatus"` command without any parameters will show the default clusterstatus output which will show the nodes configured with active ESA FM Agent cluster.

`-v` Enables a more detailed clusterstatus. Status for both ESA Master Agent and ESA FM Agent will be shown for each member/node.

4.4.7.3 Command Output

The command output presents a list of all active members in the cluster. Please note that inactive members are not visible even though they are configured to be part of the cluster. Inactive members are the members that have an ESA not running or if the entire node is down.

The following is the output format:

```
Column1  Column2   Column3 Column4  Column5    Column6
<master> <command> <esama> <esafma> <hostname> <ipaddr>
```

The following data is found in the output.

<master> This column indicates if the member is defined to act as master or not.

`M`	The cluster member is master and active.
`(M)`	The cluster member is master, but standby.
empty	The cluster member is not defined to act as master.

<command> This column indicates on which node/member the cluster status command is executed.

*	This indicator shows on which node in the cluster the status command is executed.

<esama> This column indicates if cluster mode is active for the ESA Master Agent.

`esama`	Cluster mode is active for the ESA Master Agent.
Inactive	Cluster mode is inactive for the ESA Master Agent.
Unknown	Status for the ESA Master Agent is unknown. Either the unknown node is down/disconnected while configured to be active, or the node running the command has inactive ESA Master Agent cluster.

<esafma> This column indicates if cluster mode is active for the ESA FM Agent.

`esafma`	Cluster mode is active for the ESA FM Agent.
Inactive	Cluster mode is inactive for the ESA FM Agent.
Unknown	Status for the ESA FM Agent is unknown. Either the unknown node is down/disconnected while configured to be active, or the node running the command has inactive ESA FM Agent cluster.

<hostname> This is the hostname of the cluster member.

<ipaddr> This is the IP address of the cluster member.

The following is an example output for "$esaclusterstatus":

```
Cluster status:
 M       HostA      10.1.1.1
(M)      HostB      10.1.1.2
         HostC      10.1.1.3
     *   HostD      10.1.1.4
         HostE      10.1.1.5
         HostG      10.1.1.7
         HostH      10.1.1.8

This example shows:
- HostA is the Active Master.
- HostB is also Master, but standby.
- The status command is executed on HostD.
- As HostF is not shown, we can assume that either the
  node is down or only the ESA is down.
```

The following is an example output for "$esaclusterstatus -v":

```
Cluster status:
 M       esama      esafma      HostA      10.1.1.1
(M)      esama      esafma      HostB      10.1.1.2
         Inactive   esafma      HostC      10.1.1.3
     *   esama      esafma      HostD      10.1.1.4
         esama      Unknown     HostE      10.1.1.5
         esama      esafma      HostG      10.1.1.7
         esama      esafma      HostH      10.1.1.8

This example shows:
- HostA is the Active Master.
- HostB is also Master, but standby.
- Cluster is inactive for Master Agent on HostC
- The status command is executed on HostD.
- Cluster is configured as active for ESA Fm Agent on HostE
  but it is not connected to the cluster
- As HostF is not shown, we can assume that either the
  node is down or only the ESA is down.
```

In case the ESA cluster mode is inactive, the output will simply indicate that the cluster mode is inactive.

```
Cluster mode inactive.
```

4.4.8 Dynamic cluster

In ESA 16, expansion and reduction of the cluster size can be done during runtime without any manual configurations on the already present nodes.

4.4.8.1 Cluster expansion

The cluster can be expanded by adding one or more nodes. When adding a new node the ESA will automatically update the seed-nodes list(s) on all nodes in the cluster. This means that no manual steps are required on the already present cluster members in order for them the recognize the new node as part of the cluster.

The following configurations need to be consider while preparing the new node(s):

- **mainCfg.xml**

  In order to activate cluster mode on the new node, set the `cluster/active` to *"yes"*. Please see Section 4.4.4.1 Feature Configuration, mainCfg.xml on page 24 for information about how to do the other necessary configurations in the mainCfg.xml.

- **cluster.conf**

  It is important that the same cluster configurations are made to the new node as for the already present nodes e.g.: If SSL is used in the cluster, TCP cannot be used in the node that the cluster is expanded with.

  The seed-nodes list(s) in the expanded node need to have **at least** one node that already is in the cluster and that is running. The seed-nodes can be seen as access points into the FM and MA clusters. When the expanded node has joined the cluster, the seed-nodes list(s) will automatically be synchronized through the whole cluster.

  Note: *At least* one node need to be added to the seed-nodes list(s). More access points into the cluster gives a more redundant expansion.

4.4.8.2 Cluster reduction

Removing a cluster member is done by stopping ESA on the corresponding node. Akka will automatically detect that the member has gone down and left the cluster. The removed node will automatically be removed from the seed-nodes list(s) on all the nodes in the cluster.

If the node is started again, it will re-join the cluster as long as the seed-nodes list(s) includes at least one running cluster member, and Akka will automatically update the seed-nodes list(s) on the added node to the current state.

Note: ESA will always keep at least 10 nodes in the seed-nodes lists. If the cluster consists of less than 10 nodes and the cluster size is reduced, the removed node will not be removed from the seed-nodes lists. This will not have any performance impact and it is the expected behavior.

4.4.9 Cluster troubleshooting

4.4.9.1 Trouble starting up a cluster node

When starting up a node in a cluster, if the cluster already contains one or more nodes, it is important to verify that the new node can connect to at least one of the already existing nodes. If not, the new node is going to start a cluster on its own assuming there are no other nodes in the cluster. This will end up in two different clusters that are unknown to each other. If this occurs, the new node needs to be restarted once the connection to a running node in the cluster can be established.

Checklist for starting up a cluster node:

1 Define at least one already existing node as seed-node in `cluster.conf` for the new node. The list of seed-nodes for the new node need to contain **at least** one already running node.

   Note: For more information how to configure seed-nodes see Section 4.4.4.2.1 on page 26.

2 Make sure the network connection is stable to the other nodes in the cluster before starting the new node. Running a TCP Ping to one of the configured seed-nodes could be helpful.

3 After starting the new node, it is recommended to run the `$esaclusterstatus` command. The output should show you a list of already existing cluster members including the new node.

4 If something went wrong while starting the new node, the `$esaclusterstatus` output will only show the new node itself. The new node then need to be restarted when the connection to the other nodes is stable.

4.4.9.2 Split/merge of cluster

A cluster is split when the connection to one or more cluster member(s) is lost and the lost cluster members creates clusters of their own.

If this happens the cluster with the oldest master is the cluster to keep. The other clusters tries to connect periodically. When the connection is back the other nodes will merge to the cluster. The Active Alarm List (AAL) from the active master will replace the AAL in the merging nodes.

Attention!

This means that during a cluster split/merge, alarms from the merging nodes can be lost!

4.5 Community Strings

4.5.1 Introduction

The SNMP v2c community string is like a user identification or password that allows access to the information of a device, published on the SNMP interface. An OSS sends the community string along with all its SNMP requests. If the community strings match, the device responds with the requested information. If the community strings do not match, an authentication failure trap is sent to the trap destination and the device simply discards the request and does not respond.

The ESA is installed with the default community strings *ESA-PC* and *ESA-PE*. These may be changed in the ESA. However, be aware of that modifying a community string in a SNMP agent requires that the community string(s) in the interfacing component(s) or system(s) must also be modified, in order to retain the access rights.

See the figure below, which shows the community strings used and their location in the ESA. The *ESA-PC* is configured for read only access, which means that only SNMP get operations are possible, while the *ESA-PE* is configured for read/write access, which means that both SNMP get and SNMP set operations may be performed.

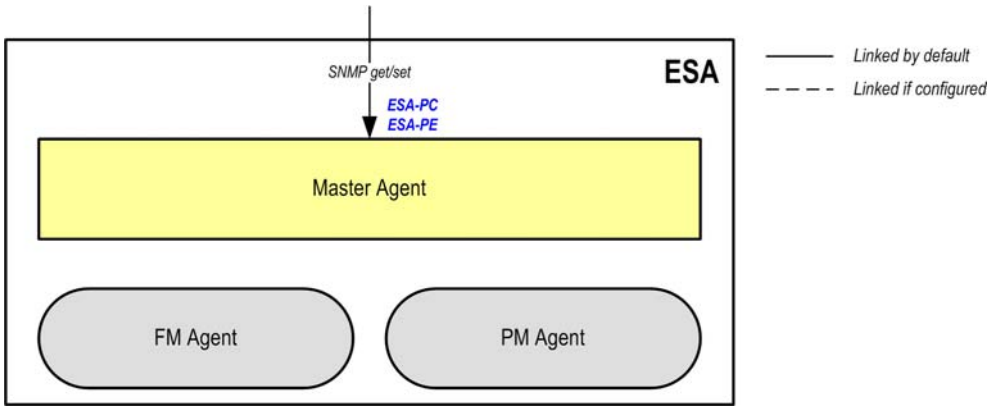

Figure 5 The SNMP v2c community strings used by default in the ESA.

For detailed information about the community string function and configuration, see MIBs SNMP-COMMUNITY-MIB and SNMP-VIEW-BASED-ACM-MIB in directory `{esa basedir}/mibs`.

4.5.2 Configuration File

The community strings are defined in configuration file `communityCfg.xml` in directory `{esa basedir}/conf`.

4.5.3 Configuration Format

The community string configurations are defined using the following format.

```
Configuration format:

<communityCfg>
  <community active="data">
    <communityString>data</communityString>
    <securityName>data</securityName>
    <contextName>data</contextName>
  </community>
</communityCfg>
```

The parameters contain the following data:

- **active**

  This attribute indicates if the defined community string is active or inactive. If set inactive, the ESA does not load the community string.

yes	The community string is active.
no	The community string is inactive.

- **communityString**

  Enter the community string to define.

- **securityName**

  Enter the security name.

  Please note that the security name must be defined in the VACM configuration. See Section 4.20 on page 75.

- **contextName** (optional)

  Enter the context name.

Examples are found in Section 4.5.5 on page 39.

4.5.4 Configuration Activation

The community string configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

4.5.5 Examples

The following is a configuration example for the ESA community strings.

```
<communityCfg>
  <community active="yes">
    <communityString>ESA-PC</communityString>
    <securityName>v1v2ReadOnlySecName</securityName>
  </community>
</communityCfg>
```

4.6 Directories

4.6.1 Overview

The ESA provides the capability to configure the directories used for both input data as well as output data.

The *input directories* are the ESA configuration directories. By default the configuration directories are placed within the deployment directory structure of the ESA, but they can be changed if needed.

The *output directories* are the directories that will be used for files created by the ESA, such as the PM Agent 3GPP XML files. By default the output directories are placed in the `/var` file system on Unix based systems, while on Windows they are placed within the deployment directory structure of the ESA.

4.6.2 Function Description

For FM and PM there are configuration directories that hold a number of configuration files. The number of files depends on how the ESA is setup for FM and PM. The files are in XML format, except for one file which is an XML Schema.

For further detail about the different configuration files for the different directories, see Reference [2] and Reference [3].

For FM, the following configuration directories are specified.

- FM Alarm Definition
- FM Alarm Translation

For PM, the following configuration directories are specified.

- PM Counter Definition
- PM Job Definition

- PM Threshold Definition

There is also an output directory used by the PM feature. The 3GPP XML files that are created is stored in the output directory.

- PM Output Directory

  The naming of the PM files and the PM file retention control is affected by the PM Agent preference configuration. See Section 4.14 on page 58.

4.6.3 Configuration File

All the ESA directories are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.6.4 Configuration Format

The directory configurations are defined using the following format.

```
Configuration format:

<directories>
  <configuration>
    <fmAlarmDefinitions>data</fmAlarmDefinitions>
    <fmAlarmTranslations>data</fmAlarmTranslations>
    <pmCounters>data</pmCounters>
    <pmJobs>data</pmJobs>
    <pmThresholds>data</pmThresholds>
  </configuration>
  <output>
    <pm3gppXml>data</pm3gppXml>
  </output>
</directories>
```

The parameters contain the following data:

- **fmAlarmDefinitions**

  Enter the configuration directory for the FM Agent Alarm Definitions.

  ```
  Default directory:
  {esa basedir}/conf/fmAlarmDefinitions/
  ```

- **fmAlarmTranslations**

  Enter the configuration directory for the FM Agent Alarm Translations.

  ```
  Default directory:
  {esa basedir}/conf/fmAlarmTranslations/
  ```

- **pmCounters**

  Enter the configuration directory for the PM Agent Counter Definitions.

  ```
  Default directory:
  {esa basedir}/conf/pmCounters/
  ```

- **pmJobs**

  Enter the configuration directory for the PM Agent Job Definitions.

  ```
  Default directory:
  {esa basedir}/conf/pmJobs/
  ```

- **pmThresholds**

  Enter the configuration directory for the PM Agent Threshold Definitions.

  ```
  Default directory:
  {esa basedir}/conf/pmThresholds/
  ```

- **pm3gppXml**

  Enter the output directory for the PM Agent 3GPP XML files.

  ```
  Default directory:
  {esa logdir}/pm3gppXml
  ```

Examples are found in Section 4.6.6 on page 42.

4.6.5 Configuration Activation

The directory configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.6.6 Examples

The following is a configuration example for the ESA directories.

```
<directories>
  <configuration>
    <fmAlarmDefinitions>/usr/local/esa/conf/
     fmAlarmDefinitions</fmAlarmDefinitions>
    <fmAlarmTranslations>/usr/local/esa/conf/
     fmAlarmTranslations</fmAlarmTranslations>
    <pmCounters>/usr/local/esa/conf/pmCounters</pmCounters>
    <pmJobs>/usr/local/esa/conf/pmJobs</pmJobs>
    <pmThresholds>/usr/local/esa/conf/pmThresholds
    </pmThresholds>
  </configuration>
  <output>
    <pm3gppXml>/var/log/esa/pm3gppXml</pm3gppXml>
  </output>
</directories>
```

4.7 FM MIB Preferences

4.7.1 Overview

The Fault Management MIBs `ERICSSON-SNF-ALARM-MIB` and `ERICSSON-SNF-EVENT-MIB` provides a couple of configurable parameters related to table sizes.

4.7.2 Function Description

4.7.2.1 Alarm Cleared List

When an active alarm has been cleared the cleared alarm is put into the Alarm Cleared List. This list is a SNMP table that contains a number of entries of cleared alarms. By default the 100 last cleared alarms are visible in the Alarm Cleared List. The number of entries is configurable.

4.7.2.2 Event History List

When an event has been sent the event is put into the Event History List. This list is a SNMP table that contains a number of entries of sent events. By default the 100 last sent events are visible in the Event History List. The number of entries is configurable.

4.7.3 Configuration File

The FM MIB parameters are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.7.4 Configuration Format

The FM MIB parameters are defined using the following format.

```
Configuration format:

<fm>
  <mib>
    <clearHistory>data</clearHistory>
    <eventHistory>data</eventHistory>
  </mib>
</fm>
```

The parameters contain the following data:

- **clearHistory**

  int32 — An integer value specifying max number of entries in the Alarm Cleared List.

  Default value: 100

- **eventHistory**

  int32 — An integer value specifying max number of entries in the Event History List.

  Default value: 100

Examples are found in Section 4.7.6 on page 44.

4.7.5 Configuration Activation

The FM MIB preferences are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.7.6 Examples

The following is a configuration example for the ESA FM MIB preferences.

```
<fm>
  <mib>
    <clearHistory>100</clearHistory>
    <eventHistory>100</eventHistory>
  </mib>
</fm>
```

4.8 Information Agent

4.8.1 Overview

The ESA Information Agent gives the OSS the possibility to read ESA and System information using SNMP get operations for use in decision making situations in relation to possible operations on the ESA as well as using the information in OSS output, such as in reports. The information is related to identification, version numbering and dates. Also, the configuration file simply contains information related to the System useful for maintenance personnel.

The system information is defined in a configuration file. The ESA information is hard coded in the ESA delivery, but the system information can be modified. The configuration file holding System information is created during the installation of the ESA, but contains by default empty parameters. This file can be edited whenever needed.

When the ESA starts the parameters are loaded and published on SNMP. The data is published according to the MIB named ERICSSON-ESA-INFORMATION-MIB. It is found in directory *{esa basedir}*/mibs.

4.8.2 Configuration File

All the System information parameters are defined in configuration file infoSystem.xml in directory *{esa basedir}*/conf.

4.8.3 Configuration Format

The following parameters are available as System information:

```
Configuration format:

<infoSystem>
  <vendor>data</vendor>
  <name>data</name>
  <nameAbb>data</nameAbb>
  <version>data</version>
</infoSystem>
```

The parameters contain the following data:

- **vendor**

  A string holding the vendor name of the system where the ESA resides.

- **name**

  A string holding the system name of the system where the ESA resides.

- **nameAbb**

  A string holding the abbreviated system name of the system where the ESA resides.

- **version**

  A string holding the version of the system where the ESA resides.

Examples are found in Section 4.8.5 on page 46.

4.8.4 Configuration Activation

The system information configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

4.8.5 Examples

The following is an example configuration for the ESA Information Agent.

```
<infoSystem>
  <vendor>Ericsson</vendor>
  <name>Ericsson Charging System</name>
  <nameAbb>CS</nameAbb>
  <version>5.0.3</version>
</infoSystem>
```

4.9 Interfaces

The ESA comes with several interfaces for FM and PM communication. The interfaces can be configured with *IP address* and *Port number*, where the IP address is optional.

The following figure shows all interfaces used in the ESA and their default values.

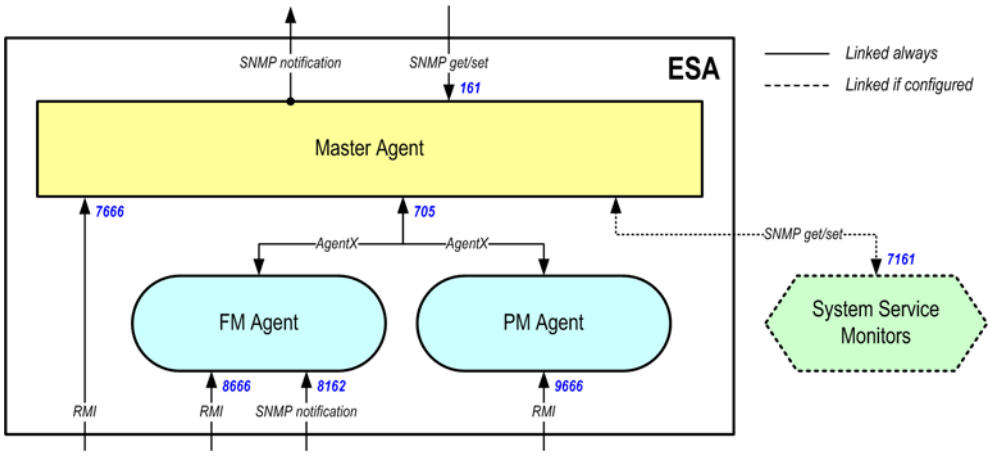

Figure 6 The ESA interfaces/ports and their default values.

The following ports are used for SNMP communication. See Section 4.12 on page 52 for details.

- **Master Agent SNMP Interface**

  Default port is “161”.

- **FM Agent SNMP Interface**

  Default port is “8162”.

The following port are used for AgentX communication. See Section 4.10 on page 48 for details.

- **Master Agent AgentX Interface**

  Default port is “705”.

The following ports are used for Java RMI communication. See Section 4.11 on page 50 for details.

- **Master Agent RMI Interface**

  Default port is “7666”.

  Note: This interface is experimental only!

- **FM Agent RMI Interface**

  Default port is “8666”.

- **PM Agent RMI Interface**

  Default port is “9666”.

The following port is used for the SSM component.

- **System Service Monitors SNMP Interface**

  Default port is “7161”.

  Note: The configuration of the SSM port is not found in this document. The port number is by default set by SSM to 161, but set to 7161 during the SSM installation in the ESA context. The port number for SSM can be changed to any port. See the SSM documentation for configuring the SSM ports.

4.10 Interface AgentX

4.10.1 Introduction

See the overview description of the ESA interfaces in Section 4.9 on page 46.

AgentX is a communication protocol used between the ESA Master Agent and subagents with AgentX capabilities.

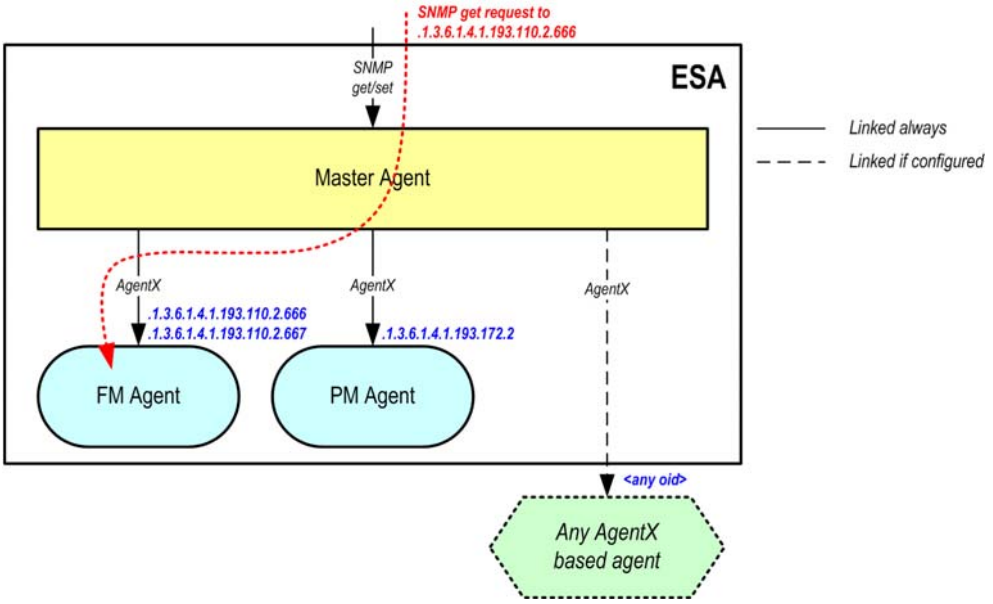

Figure 7 Based on the OID in the SNMP get/set request, the Master Agent uses AgentX to route the request to the correct subagent.

In the ESA the AgentX is by default active and used between the ESA Master Agent and the subagents FM Agent and PM Agent.

Any agent with AgentX capabilities can connect to the ESA Master Agent. The AgentX capability in the ESA Master Agent is always active and can not be turned off.

For detailed information about the AgentX function and configuration, see MIB AGENTX-MIB in directory *{esa basedir}*/mibs.

There is one AgentX interface to configure:

- Master Agent

  This interface is used by the subagents with AgentX capabilities.

The purpose of configuring IP addresses and ports for the AgentX interfaces is to bind the ESA to use a limited number of interfaces in a system with multiple interfaces.

If an IP address is specified, the ESA will only run AgentX on the port specified on the IP address specified.

If no IP address is specified, the ESA will run AgentX on the port specified on all IP addresses available in the system.

If a loopback IP address is used, the ESA only accepts AgentX connections from subagents running locally on the same node.

4.10.2 Configuration File

The AgentX parameters are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.10.3 Configuration Format

The AgentX interface configuration is defined using the following format.

```
Configuration format:

<agentx>
  <ip>data</ip>
  <port>data</port>
</agentx>
```

The parameters contain the following data:

- **ip**

  Specify the IP address to bind the AgentX interface to.

  Default: ***127.0.0.1***

  If bind to IP address:

  ```
  <ip>19.72.9.21</ip>
   <port>port</port>
  ```

  If bind to all available IP addresses:

  ```
  <ip/>
  ```

```
<port>port</port>
```

- **port**

  Enter the port number to use for the AgentX interface.

  Default: **705**

Examples are found in Section 4.10.5 on page 50.

4.10.4 Configuration Activation

The AgentX configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.10.5 Examples

The following is an example AgentX interface configuration.

```
<agentx>
  <ip>19.72.9.21</ip>
  <port>705</port>
</agentx>
```

4.11 Interface Java RMI

4.11.1 Introduction

See the overview description of the ESA interfaces in Section 4.9 on page 46.

There are three RMI interfaces to configure:

- Master Agent

  This interface is used to access the Master Agent API.

  Note: Experimental only!

- FM Agent

  This interface is used to access the FM Agent API.

- PM Agent

  This interface is used to access the PM Agent API.

The purpose of configuring IP addresses and ports for the RMI interfaces is to bind the ESA to use a limited number of interfaces in a system with multiple interfaces.

If an IP address is specified, the ESA will only run RMI on the port specified on the IP address specified.

If no IP address is specified, the ESA will run RMI on the port specified on all IP addresses available in the system.

If a loopback IP address is used, the ESA only accepts RMI connections from API users running locally on the same node.

4.11.2 Configuration File

The Java RMI parameters are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.11.3 Configuration Format

The RMI interface configurations are defined using the following format.

```
Configuration format:

<rmi>
  <ma>
    <ip>data</ip>
    <port>data</port>
  </ma>
  <fm>
    <ip>data</ip>
    <port>data</port>
  </fm>
  <pm>
    <ip>data</ip>
    <port>data</port>
  </pm>
</rmi>
```

The parameters contain the following data:

- **ip**

  Specify the IP address to bind the RMI interface to.

  Default: ***127.0.0.1***

  If bind to IP address:

  ```
  <ip>19.72.9.21</ip>
  ```

```
  <port>port</port>
```

If bind to all available IP addresses:

```
 <ip/>
  <port>port</port>
```

- **port**

  Enter the port number to use for the RMI interface.

  Default Master Agent: `7666`

  Default FM Agent: `8666`

  Default PM Agent: `9666`

Examples are found in Section 4.11.5 on page 52.

4.11.4 Configuration Activation

The interfaces configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.11.5 Examples

The following is an example RMI interface configuration.

```
<rmi>
  <ma>
    <ip/>
    <port>7666</port>
  </ma>
  <fm>
    <ip>2001:0db8:85a3:0042:1000:8a2e:0370:7334</ip>
    <port>8666</port>
  </fm>
  <pm>
    <ip>127.0.0.1</ip>
    <port>9666</port>
  </pm>
</rmi>
```

4.12 Interface SNMP

4.12.1 Introduction

See the overview description of the ESA interfaces in Section 4.9 on page 46.

There are two SNMP interfaces to configure:

- Master Agent

  This interface is used by the OSS accessing the ESA.

- FM Agent

  This interface is used by SNMP subagents for sending SNMP traps and notifications to the FM Agent for alarm translation.

The purpose of configuring IP addresses and ports for the SNMP interfaces is to bind the ESA to use a limited number of interfaces in a system with multiple interfaces. Also, specifying an IP address also provides the ESA with the sender IP address to use when sending SNMP notifications.

If one or several IP addresses are specified, the ESA will only listen to the port specified on the IP addresses specified. The first specified IP address in the list of IP addresses will be used as the sender IP address in SNMP notifications sent from the ESA.

If no IP address is specified, the ESA will listen to the port specified on all IP addresses available in the system. The system default IP address will be used as the sender IP address.

If a loopback IP address is used, the ESA only accepts SNMP traffic from locally running SNMP components.

4.12.2 Configuration File

The SNMP interface parameters are defined in configuration file `mainCfg.xml` in directory *`{esa basedir}`*`/conf`.

4.12.3 Configuration Format

The SNMP interface configurations are defined using the following format.

```
Configuration format:

<snmp>
  <masterAgent>
    <ip>data</ip>
    <port>data</port>
  </masterAgent>
  <fmAgent>
    <ip>data</ip>
    <port>data</port>
  </fmAgent>
</snmp>
```

The parameters contain the following data:

- **ip**

  Specify all IP addresses to bind the SNMP interface to.

  Default for Master Agent: ***(empty)***

  Default for FM Agent: ***127.0.0.1***

  If bind to only one IP address:

  ```
  <ip>19.72.9.21</ip>
   <port>port</port>
  ```

  If bind to three IP addresses:

  ```
  <ip>10.1.1.1</ip>
   <ip>170.45.5.100</ip>
   <ip>19.72.9.21</ip>
   <port>port</port>
  ```

  If bind to all available IP addresses, which also means "no binding":

  ```
  <ip/>
   <port>port</port>
  ```

- **port**

  Enter the port number to use for the SNMP interface.

  Default Master Agent: **161**

  Default FM Agent: **8162**

Examples are found in Section 4.12.5 on page 54.

4.12.4 Configuration Activation

The interface configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.12.5 Examples

The following is an example SNMP interface configuration.

```
Example 1: Bind Master Agent to one IPv4 address and
one IPv6 address.

<snmp>
  <masterAgent>
    <ip>19.72.9.21</ip>
    <ip>2001:0db8:85a3:0042:1000:8a2e:0370:7334</ip>
    <port>161</port>
  </masterAgent>
  <fmAgent>
    <ip>127.0.0.1</ip>
    <port>8162</port>
  </fmAgent>
</snmp>
```

The Example 1 above will bind to only two IP addresses in the system. This means that communication towards the ESA is done using any of the two IP addresses. Other IP addresses in the system will not be used by the ESA. The IP address 19.72.9.21, which is specified first in the list of IP addresses, will be used as the sender IP address.

```
Example 2: Do not bind Master Agent to any IP address, but
select one IPv4 address to be the sender IP address.

<snmp>
  <masterAgent>
    <ip>19.72.9.21</ip>
    <ip></ip>
    <port>161</port>
  </masterAgent>
  <fmAgent>
    <ip>127.0.0.1</ip>
    <port>8162</port>
  </fmAgent>
</snmp>
```

The Example 2 above will not bind to any IP address in the system as the empty <ip> element indicates "binding to all". This means that communication towards the ESA is done using any IP address in the system. However, the IP address 19.72.9.21, which is specified first in the list of IP addresses, is specifically chosen to act as the sender IP address.

4.13 Java Virtual Machine

4.13.1 Memory Settings

4.13.1.1 ESA Services

The Java heap size can be tuned, if needed. The heap size is configured by using the following parameters.

-Xms Initial heap size.

Default value: 128 MB

-Xmx Maximum heap size.

Default value for FM Agent: 512 MB

Default value for Master Agent and PM Agent: 128 MB

To modify the Java heap size for the ESA services perform the following steps:

1. Open the java configuration file `*.vmoptions` in a text editor for the ESA service to modify. The file is located in *{esa basedir}*`/bin`.
2. Edit the following lines:

   For FM Agent:
   ```
   -Xms128m
   -Xmx512m
   ```

   For Master Agent and PM Agent:
   ```
   -Xms128m
   -Xmx128m
   ```

   and set the wanted memory size values.
3. Save the changes, exit the text editor and restart the ESA.

4.13.1.2 ESA CLI Commands

The ESA CLI commands are executed as small java processes. They are executed for a short time period, but in some cases on some platforms it has been observed that a huge amount of memory has been reserved while executing the commands. This will most likely happen on servers with lot of RAM installed (> 64 GB). In case a limitation of RAM usage is needed for a CLI command, the same method as described in Section 4.13.1.1 on page 56 can be used, but new `.vmoptions` files must be created as there are no such files by default.

1. Create a file `.vmoptions` for the CLI command that you want to control the RAM.

Example) If a RAM usage limit is needed for command `pmwritecounter`, then create file `pmwritecounter.vmoptions`.

2. Add the following parameters `-Xms` and `-Xmx` to the file. See Section 4.13.1.1 on page 56 for parameter descriptions.

   Example)

   ```
   -Xms64m
   -Xmx64m
   ```

3. Save the changes and exit the text editor.

4.13.2 CPU Usage Control

The ESA CLI commands are executed as small java processes. They are executed for a short time period, but in some cases on some platforms it has been observed that they are executed with high CPU load during the execution of the commands.

Please note that it is not possible to control the CPU usage of a java process by using a vmoptions, similar to controlling RAM. Instead OS commands should be used.

The use of CLI command `taskset` has been successfully used in a Linux environment.

```
Example)

To bind the ESA command 'pmwritecounter' to processor #0:

# taskset 01 pmwritecounter <module> <counter> <value> <time>

where 'module', 'counter', 'value' and 'time' are arguments
to the command 'pmwritecounter'.
```

Please see the OS documentation for the platform being used in order to limit the CPU usage in your environment.

4.13.3 IP Address Cache Time

If the ESA resides on a system where the communication to a subagent is setup using a hostname that changes over time, there is a need to refresh the IP address of the hostname now and then. When the ESA is started, the IP address of the hostname is by the ESA read from a DNS and the IP address is cached. The default setting for cache time is ‘‘forever’’. This means that once the ESA reads the IP address for a hostname, that IP address is used until the ESA is restarted.

To configure the ESA to refresh the IP address on interval perform the following steps:

1. Open the java configuration file `*.vmoptions` in a text editor for the ESA service to modify. The file is located in *{esa basedir}*`/bin`.

2. Append the following line to the file:

   `-Dsun.net.inetaddr.ttl=<`*time*`>`

   where `<time>` is equal to the refresh time in seconds.

3. Save the changes, exit the text editor and restart the ESA.

4.13.4 Dual Stack IP

On dual stack machines with both IPv4 and IPv6 addresses, there might be issues with the ESA startup time. In order to shorten the startup time the solution could be to mark IPv4 is preferred. This is done by setting the java property `java.net.preferIPv4Stack`.

To configure the ESA to prefer IPv4 addresses perform the following steps:

1. Open the java configuration file `*.vmoptions` in a text editor for the ESA service to modify. The file is located in *{esa basedir}*`/bin`.

2. Append the following line to the file:

   `-Djava.net.preferIPv4Stack=true`

3. Save the changes, exit the text editor and restart the ESA.

4.14 PM Agent Preferences

4.14.1 Overview

The PM Agent has a few configuration parameters regarding the creation of 3GPP XML files.

The parameter *Unique Identity* is about naming the output files. Each 3GPP XML file will be "tagged" with the string used in the `uniqueId` field.

The parameters *File Retention Period* and *File Retention Scan* are about managing the growing number of 3GPP XML output files.

4.14.2 Function Description

The 3GPP XML files are created according to the job definitions in each interval specified. The number of files will constantly grow, which over time might fill up the disk space. Using file retention control the number of files are limited by controlling the age of the files.

It is recommended to have an OSS reading the PM output files and then delete them when successfully read. But, if there are no such process reading/deleting files, the disk usage can be controlled using ESA internal disk usage control processes.

4.14.3 Configuration File

The PM Agent parameters are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.14.4 Configuration Format

The PM Agent preferences parameters are defined using the following format.

```
Configuration format:

<pm>
  <uniqueId>data</uniqueId>
  <fileRetentionPeriod>data</fileRetentionPeriod>
  <fileRetentionScan>data</fileRetentionScan>
</pm>
```

The following parameters are defined:

- `uniqueId`

  This is a string used to represent the node that runs the ESA.

  Valid characters are A to Z, a to z and 0 to 9 using no spaces. Also characters "-" and "_" are allowed. The length is restricted to 2-40 characters.

  Default string: `ESA`

  The unique identity is used at the following location:

  – 3GPP XML Filename

    The unique identity is appended to each 3GPP XML filename according to the following format.

    `A20140921.1500-1505-MYJOB_<uniqueid>.xml`

    If the unique identity is set to `MYSYSTEM`, the file name will be the following.

    `A20140921.1500-1505-MYJOB_MYSYSTEM.xml`

    Please note that using characters "-" or "_" in the unique identity string, such as `MY-SYSTEM` or `MY_SYSTEM` will lead to the following filenames.

```
A20140921.1500-1505-MYJOB_MY-SYSTEM.xml
A20140921.1500-1505-MYJOB_MY_SYSTEM.xml
```

As you can see the characters "-" and "_" might be misleading since there are already such characters used to separate other data in the filename. Please try to avoid using the characters "-" and "_".

Note that the string `MYJOB` in the example filename is the *Job Identity*, which is configurable and described in Reference [3].

- `fileRetentionPeriod`

  The period in *days* that the PM output files are kept on the file system. If the files are older than the specified number of days, the ESA will delete them.

  This parameter is used for having control of the PM Agent output directory disk usage. Files that are older than the specified period are deleted from the file system in order to not fill up the system storage.

  Use value "`0`" to deactivate the file retention control.

  Examples: Use "`4`" for four days or "14" for two weeks.

- `fileRetentionScan`

  The interval in *minutes*, which tells the ESA how often the file retention control disk usage scan shall be performed. At each interval the PM Agent performs a scan of the PM output directory and deletes all the files older than the defined period in parameter `fileRetentionPeriod`.

  A short interval makes the ESA scan the disk usage more often, which in turn more often puts a load on the system. Edit this parameter to fine tune the ESA process load if the system often is working in high load. If the number of files in the PM output directory is high this value should be kept high or the file retention control should be turned off.

  Default value at ESA installation is 30 minutes. The lowest possible value is 10 minutes. The scan is deactivated if parameter `fileRetentionPeriod` is set to "`0`".

  Example: Use "`20`" for 20 minutes or "`240`" for four hours.

Examples are found in Section 4.14.6 on page 60.

4.14.5 Configuration Activation

The PM Agent configurations are only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.14.6 Examples

The following is a configuration example for the PM Agent settings.

```
<pm>
  <uniqueId>ESA</uniqueId>
  <fileRetentionPeriod>14</fileRetentionPeriod>
  <fileRetentionScan>60</fileRetentionScan>
</pm>
```

4.15 SNMP Proxy

4.15.1 Introduction

The SNMP proxy capability in the ESA Master Agent is used for routing SNMP operations, such as SNMP-GET and SNMP-SET, to SNMP subagents.

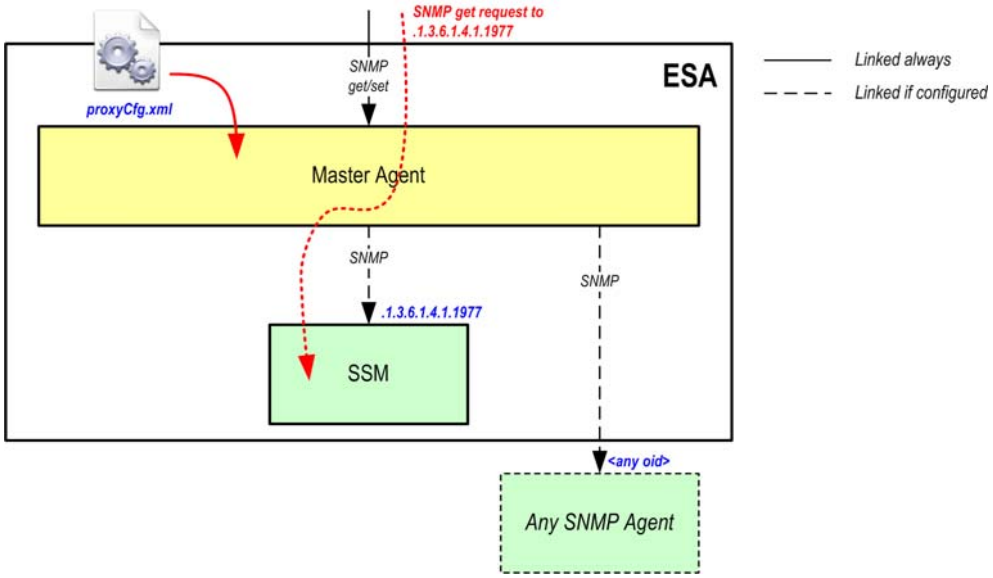

Figure 8 Based on the OID in the SNMP get/set request, the Master Agent uses the proxy configuration to route the request to the correct subagent.

The figure above shows the OIDs defined for a default ESA installation; the SSM OIDs. Please note that the SSM proxy settings by default are inactive. If SSM is in operation, the predefined SSM proxy settings can be activated by changing configuration parameter `active` from `no` to `yes`.

Any SNMP agent which allows SNMP get/set requests may be connected to the Master Agent by providing SNMP route information to the Master Agent proxy configuration file. The red dotted line in the figure above shows an example of a SNMP get request, sent to the OID `.1.3.6.1.4.1.1977`. The

proxy configuration settings tells the Master Agent to route the SNMP message to subagent SSM.

For detailed information about the SNMP Proxy function and configuration, see MIBs SNMP4J-PROXY-MIB and SNMP-TARGET-MIB in directory *{esa basedir}*/mibs.

4.15.2 Configuration File

The SNMP Proxy parameters are defined in configuration file proxyCfg.xml in directory *{esa basedir}*/conf.

4.15.3 Configuration Format

The SNMP Proxy configurations are defined using the following format.

Configuration format:

```
<proxyCfg>
  <proxyDefinition active="data">
    <proxy proxyType="data">
      <subtree>data</subtree>
      <contextName>data</contextName>
    <proxy>
    <target snmpVersion="data">
      <ip>data</ip>
      <port>data</port>
      <securityName>data</securityName>
      <securityLevel>data</securityLevel>
      <timeout>data</timeout>
      <retries>data</retries>
    </target>
  </proxyDefinition>
</proxyCfg>
```

The parameters contains the following data:

- **proxyDefinition/active**

  This attribute indicates if the proxy definition is active or not.

yes	The proxy definition is active.
no	The proxy definition is inactive.

- **proxyDefinition/proxy/proxyType**

  This attribute indicates the proxy access type.

`readOnly`	Forward read only operations only.
`readAndWrite`	Forward both read and write operations.
`noProxy`	Forwarding inactive.

- **`proxyDefinition/proxy/subtree`**

  The part of the MIB tree implemented at the configured address (subagent).

- **`proxyDefinition/proxy/contextName`** (optional)

  The context name.

- **`proxyDefinition/target/snmpVersion`**

  This attribute indicates which SNMP version to use to the subagent.

`v2c`	Target is using SNMP v2c.
`v3`	Target is using SNMP v3.

- **`proxyDefinition/target/ip`**

  The IP address of the subagent.

- **`proxyDefinition/target/port`**

  The port number of the subagent.

- **`proxyDefinition/target/securityName`**

  The community string or security name of the subagent.

  Note: If security name is used also USM must be setup. See Section 4.19 on page 72.

- **`proxyDefinition/target/securityLevel`**

  The security level of the subagent.

`noAuthNoPriv`	No Authentication and no Privacy used.
`authNoPriv`	Authentication used, but no Privacy.
`authPriv`	Authentication and Privacy used.

  Note: If SNMPv2c is used, security level `noAuthNoPriv` should be used.

- **`proxyDefinition/target/timeout`** (Optional)

  The maximum round trip time in seconds for communicating with the subagent.

If not specified the default value “2 seconds” is used.

- **`proxyDefinition/target/retries`** (Optional)

  The number of retries to be attempted communicating with the subagent when a response is not received.

  If not specified the default value “1 retry” is used.

Examples are found in Section 4.15.5 on page 64.

4.15.4 Configuration Activation

The SNMP proxy configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

4.15.5 Examples

The following example shows a proxy configuration for a SNMP subagent using SNMPv2c.

```
<proxyCfg>
  <proxyDefinition active="yes">
    <proxy proxyType="readAndWrite">
      <subtree>1.3.6.1.4.1.1977</subtree>
    <proxy>
    <target snmpVersion="v2c">
      <ip>127.0.0.1</ip>
      <port>7161</port>
      <securityName>private</securityName>
      <securityLevel>noAuthNoPriv</securityLevel>
      <timeout>5</timeout>
      <retries>3</retries>
    </target>
  </proxyDefinition>
</proxyCfg>
```

4.16 SNMPv3

4.16.1 Overview

This chapter explains how to enable SNMPv3 in the Ericsson SNMP Agent on a very high level. It is assumed that the reader/user of this chapter is familiar with SNMPv3 and knows the configurations and concepts needed.

The overall procedure is to set up the USM configuration, VACM configuration, Trap Destination configuration and maybe also Proxy configuration according to the following process.

1. One or several USM users are setup, which are setup with a number of security settings.

   See Section 4.19 on page 72.

2. One or several VACM views and access rights are created, which are linked to the USM user(s).

   See Section 4.20 on page 75.

3. One or several Trap Destinations can be created, which in turn also are linked to the USM user(s).

   See Section 4.18 on page 67.

4. If there are subagents connected to the ESA also Proxies can be setup, which are linked to USM and VACM settings.

   See Section 4.15 on page 61.

4.16.2 Boot Counter File

This section is for information only. The files described in this section shall not be altered or removed.

The ESA maintains two boot counter files, which are used by the SNMPv3 function according to RFC 3414. The following two files are found on the file system where the ESA is deployed.

- `<esa basedir>/bin/esama.bc`

  The Master Agent boot counter file.

- `<esa basedir>/bin/esafma.bc`

  The FM Agent boot counter file.

4.17 Subagent AAL Synchronization

4.17.1 Introduction

The Subagent AAL Synchronization (SAS) feature is an API that provides an integrator with the possibility to synchronize alarms in subagents with the ESA during ESA startup.

Please see Reference [2] for further information about this function.

4.17.2 Configuration File

The SAS parameters are defined in configuration file `mainCfg.xml` in directory *{esa basedir}*`/conf`.

4.17.3 Configuration Format

The SAS configuration is defined using the following format.

Configuration format:

```
<sas active="data">
  <agentHost>data</agentHost>
  <agentPort>data</agentPort>
  <agentName>data</agentName>
  <timeout>data</timeout>
</sas>
```

The attribute contains the following data:

- **sas/active**

  This attribute indicates if the proxy definition is active or not.

`yes`	The SAS function is active.
`no`	The SAS function is inactive.

The elements contain the following data:

- **sas/agentHost**

  This is the IP address of the SAS agent RMI server.

- **sas/agentPort**

  This is the port number to the SAS agent RMI server.

- **sas/agentName**

  This is the RMI name of the SAS agent.

- **sas/timeout**

  This is the timer value (defined in seconds) controlling the process for communicating with the SAS server. If the time runs out before the synchronization with the SAS server is complete, the ESA will stop the synchronization. The timer is used for not hanging the ESA in case the SAS server has a too long process time or even hangs.

Examples are found in Section 4.17.5 on page 67.

4.17.4 Configuration Activation

The SAS configuration is only read at ESA startup, which means that the ESA must be started or restarted to make the changes take effect.

4.17.5 Examples

The following is a configuration example for the SAS feature.

```
<sas active="yes">
  <agentHost>127.0.0.1</agentHost>
  <agentPort>12345</agentPort>
  <agentName>mySasServer</agentName>
  <timeout>15</timeout>
</sas>
```

4.18 Trap Destination

4.18.1 Overview

The trap destination tells the ESA to which OSS to send the SNMP notifications.

The Master Agent in the ESA is responsible for handling the trap destination(s). The figure below shows how the trap destinations are defined in the ESA.

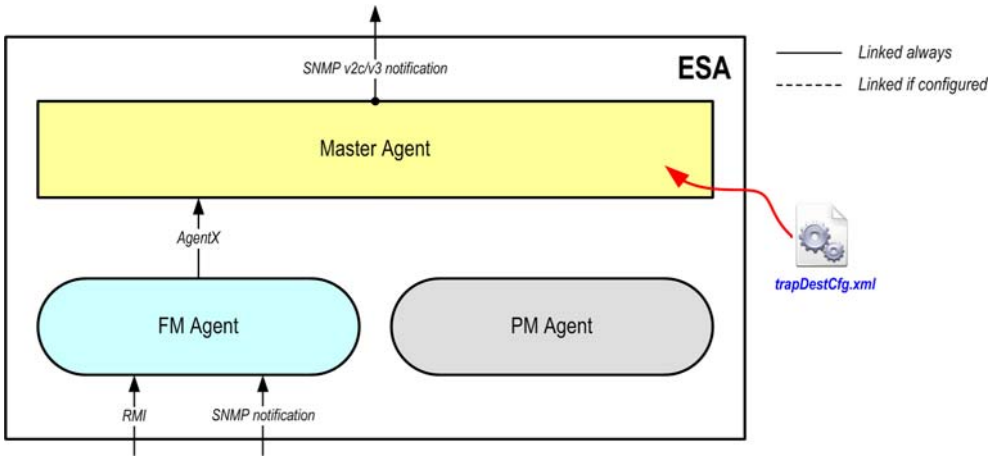

Figure 9 An overview of the ESA trap destination configurations.

4.18.2 Function Description

The FM Agent is the module triggering the SNMP notifications from the ESA on incoming order from the RMI and/or SNMP interfaces. The Master Agent is the module being responsible for managing the interface to the OSS.

The ESA supports handling of single or multiple OSSes. At least one trap destination must be defined to get the ESA to send SNMP notifications. The ESA can run with no trap destinations defined at all, which simply means that no SNMP notifications are sent from the ESA, but all other ESA features will work as normal.

The status of each specified trap destination can easily be controlled by a parameter. Each trap destination can, independently of another, be made inactive without removing the trap destination configuration. This is useful if there is a need to temporarily stop the sending of alarms to a manager.

If a trap destination is to be setup using SNMPv3 format, please see the Section 4.16 on page 64 for additional configurations needed.

For detailed information about the Trap Destination function and configuration, see MIBs SNMP-NOTIFICATION-MIB och SNMP-TARGET-MIB in directory *{esa basedir}*/mibs. The MIBs VACM-MIB (setting up view and access) and SNMP-NOTIFICATION-LOG-MIB (notification history) could be of interest.

If the ESA is running in cluster mode, it is recommended to **not** define trap destinations for the ESAs that are not setup as masters. Any ESA will at start/restart send a coldstart trap. In a cluster the ESAs that are not running as master should not send any traps at all.

4.18.3 Configuration File

The trap destination parameters are defined in configuration file trapDestCfg.xml in directory *{esa basedir}*/conf.

4.18.4 Configuration Format

The following parameters are available for a trap destination.

```
Configuration format:

<trapDestCfg>
  <managerDefinition snmpVersion="data" active="data">
    <ip>data</ip>
    <port>data</port>
    <securityName>data</securityName>
    <securityLevel>data</securityLevel>
  </managerDefinition>
</trapDestCfg>
```

The configuration attributes contain the following data:

- **trapDestCfg/managerDefinition/active**

`yes`	The trap destination definition is active.
`no`	The trap destination definition is inactive.

This attribute specifies if the trap destination configuration is active (`yes`) or inactive (`no`). By setting it to inactive the ESA will disregard reading the configuration and thus not send any traps to that destination. This setting is useful since the configuration can be kept as is while it is possible to stop the traps by only deactivating it with a ‘‘no’’.

- **`trapDestCfg/managerDefinition/snmpVersion`**

`v2c`	The traps are sent as v2c traps.
`v3`	The traps are sent as v3 traps.

This attribute specifies if the traps to the defined destination shall be sent as v2c traps (`v2c`) or v3 traps (`v3`). Please keep in mind that the option `v3` requires additional SNMPv3 configuration. See Section 4.16 on page 64.

The configuration elements contain the following data:

- **`trapDestCfg/ip`**

  This element contains the trap destination IP address or Hostname.

- **`trapDestCfg/port`**

  This element contains the trap destination port number.

- **`trapDestCfg/securityName`**

  This element contains the security name used for this destination configuration.

- **`trapDestCfg/securityLevel`**

  This element contains the security level used for this destination configuration.

`noAuthNoPriv`	No Authentication and no Privacy used.
`authNoPriv`	Authentication used, but no Privacy.
`authPriv`	Authentication and Privacy used.

Note: If SNMPv2c is used, security level `noAuthNoPriv` should be used.

Examples are found in Section 4.18.6 on page 70.

4.18.5 Configuration Activation

The trap destination configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

4.18.6 Examples

4.18.6.1 Example 1: SNMPv2c

The following example shows how the trap destination can be setup using Trap Destination configuration for SNMPv2c traps including VACM configuration. USM configuration is not applicable for SNMPv2c.

```
File: trapDestCfg.xml:

<trapDestCfg>
  <managerDefinition snmpVersion="v2c" active="yes">
    <ip>100.12.34.56</ip>
    <port>162</port>
    <securityName>v1v2ReadWriteSecName</securityName>
    <securityLevel>noAuthNoPriv</securityLevel>
  </managerDefinition>
</trapDestCfg>
```

```
File: vacmCfg.xml:

<securityToGroup active="yes">
  <securityModel>SNMPv2</securityModel>
  <securityName>v1v2ReadWriteSecName</securityName>
  <groupName>v1v2ReadWriteGroup</groupName>
</securityToGroup>

<access active="yes">
  <groupName>v1v2ReadWriteGroup</groupName>
  <securityModel>SNMPv2</securityModel>
  <securityLevel>noAuthNoPriv</securityLevel>
  <contextPrefix/>
  <contextMatch>exact</contextMatch>
  <readViewName>fullReadView</readViewName>
  <writeViewName>fullWriteView</writeViewName>
  <notifyViewName>fullNotifyView</notifyViewName>
</access>

<view active="yes">
  <name>fullNotifyView</name>
  <subtree>1.3</subtree>
  <mask/>
  <type>included</type>
</view>
```

```
File: usmCfg.xml

No configuration for SNMPv2c traps!
```

The example above shows a SNMPv2c trap destination with security name `v1v2ReadWriteSecName` being defined. The security name `v1v2ReadWriteSecName` points at access group `v1v2ReadWriteGroup`, which in turn points at notify view `fullNotifyView`. The notify view `fullNotifyView` is set to allow all traps that comply to OID "1.3" to be sent.

For further information about VACM configuration, see Section 4.20 on page 75.

4.18.6.2 Example 2: SNMPv3

The following example shows how the trap destination can be setup using Trap Destination configuration for SNMPv3 traps including VACM and USM configurations.

```
File: trapDestCfg.xml:

<trapDestCfg>
  <managerDefinition snmpVersion="v3" active="yes">
    <ip>100.12.34.56</ip>
    <port>162</port>
    <securityName>v3AuthSHAPrivDESSecName</securityName>
    <securityLevel>authPriv</securityLevel>
  </managerDefinition>
</trapDestCfg>
```

```
File: vacmCfg.xml

<securityToGroup active="yes">
  <securityModel>USM</securityModel>
  <securityName>v3AuthSHAPrivDESSecName</securityName>
  <groupName>v3AuthPrivGroup</groupName>
</securityToGroup>

<access active="yes">
  <groupName>v3AuthPrivGroup</groupName>
  <securityModel>USM</securityModel>
  <securityLevel>authPriv</securityLevel>
  <contextPrefix/>
  <contextMatch>exact</contextMatch>
  <readViewName>fullReadView</readViewName>
  <writeViewName>fullWriteView</writeViewName>
  <notifyViewName>fullNotifyView</notifyViewName>
</access>

<view active="yes">
  <name>fullNotifyView</name>
  <subtree>1.3</subtree>
  <mask/>
  <type>included</type>
</view>
```

```
File: usmCfg.xml

<usmUser active="yes">
  <securityName>v3AuthSHAPrivDESSecName</securityName>
  <authProtocol>AuthSHA</authProtocol>
  <authPassword>myAuthPassword</authPassword>
  <privProtocol>PrivDES</privProtocol>
  <privPassword>myPrivPassword</privPassword>
</usmUser>
```

The example above shows a SNMPv3 trap destination with security name `v3AuthSHAPrivDESSecName` being defined. The security name `v3AuthSHAPrivDESSecName` points at access group `v3AuthPrivGroup`, which in turn points at notify view `fullNotifyView`. The notify view `fullNotifyView` is set to allow all traps that comply to OID "1.3" to be sent. In order to allow sending of traps with both authentication and privacy control, USM is defined for the security name `v3AuthSHAPrivDESSecName` with both authentication and privacy settings.

For further information about VACM configuration, see Section 4.20 on page 75, and for USM configuration, see Section 4.19 on page 72.

4.19 USM

4.19.1 Introduction

The ESA fully supports RFC 3415, which describes USM.

See the MIB SNMP-USER-BASED-SM-MIB for further information.

4.19.2 Configuration File

The USM in the ESA is configured in configuration files `usmCfg.xml`.

Please note that there are default configurations created for USM. They are however by default inactive.

4.19.3 Configuration Format

The USM user is setup following the MIB SNMP-USER-BASED-SM-MIB. The user is setup with authentication and privacy settings.

The following parameters are to be defined:

```
Configuration format:

<usmCfg>
  <usmUser active="data">
    <securityName>data</securityName>
    <authProtocol>data</authProtocol>
    <authPassword>data</authPassword>
    <privProtocol>data</privProtocol>
    <privPassword>data</privPassword>
  </usmUser>
</usmCfg>
```

The parameters contain the following data:

- **usmCfg/usmUser/active**

  This attribute indicates if the USM user definition is active or not.

`yes`	The USM user definition is active.
`no`	The USM user definition is inactive.

- **usmCfg/usmUser/securityName**

  The name of the USM user.

- **usmCfg/usmUser/authProtocol**

An authentication protocol to use for the USM user.

`AuthMD5`	Authentication protocol MD5 is used.
`AuthSHA`	Authentication protocol SHA is used.

- **`usmCfg/usmUser/authPassword`**

  An authentication password to use for the USM user.

- **`usmCfg/usmUser/privProtocol`**

  A privacy protocol to use for this USM user.

`PrivDES`	Privacy protocol DES is used.
`PrivAES128`	Privacy protocol AES128 is used.
`PrivAES192`	Privacy protocol AES192 is used.
`PrivAES256`	Privacy protocol AES256 is used.

  Note: In order to use `PrivAES192` or `PrivAES256`, the Java Cryptography Extension (JCE) must be installed in the Java Runtime Environment that is used by the ESA.

- **`usmCfg/usmUser/privPassword`**

  A privacy password to use for the USM user.

Examples are found in Section 4.19.5 on page 74.

4.19.4 Configuration Activation

The USM configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

4.19.5 Examples

The following example shows the authentication and privacy settings for the security name `v1v2ReadOnlySecName`.

```
<usmCfg>
  <usmUser active="yes">
    <securityName>v1v2ReadOnlySecName</securityName>
    <authProtocol>AuthMD5</authProtocol>
    <authPassword>myAuthPassword</authPassword>
    <privProtocol>PrivAES128</privProtocol>
    <privPassword>myPrivPassword</privPassword>
  </usmUser>
</usmCfg>
```

4.20 VACM

4.20.1 Introduction

The ESA fully supports RFC 3416, which describes VACM.

See the MIB SNMP-VIEW-BASED-ACM-MIB for further information.

4.20.2 Configuration File

The VACM in the ESA is configured in configuration file `vacmCfg.xml`.

Please note that there are default configurations created for VACM for v1, v2c and v3. The configurations for SNMP v1 and SNMP v2c are active, but for SNMPv3 inactive

4.20.3 Configuration Format

4.20.3.1 Main Structure

The VACM is setup following the MIB SNMP-VIEW-BASED-ACM-MIB. The VACM is about creating views and setting access rights to the views.

The VACM configuration is built up in three main sections.

Configuration format:

```
<vacmCfg>
  <securityToGroup>
     ...
  </securityToGroup>
  <accesses>
     ...
  </accesses>
  <views>
     ...
  </views>
</vacmCfg>
```

The sections are defined as following:

- **vacmCfg/securityToGroup**

  This element maps a combination of securityModel and securityName into a groupName which is used to define an access control policy for a group of principals.

- **vacmCfg/accesses**

This element holds the access rights for groups.

- **vacmCfg/views**

  This element holds information on a particular family of view subtrees included in or excluded from a particular SNMP context's MIB view.

4.20.3.2 Element: SecurityToGroup

The element `securityToGroup` contains the following parameters:

Configuration format:

```
<securityToGroup active="data">
  <securityModel>data</securityModel>
  <securityName>data</securityName>
  <groupName>data</groupName>
</securityToGroup>
```

The parameters contain the following data:

- **securityToGroup/active**

  This attribute indicates if the VACM securityToGroup definition is active or not.

`yes`	The VACM SecurityToGroup definition is active.
`no`	The VACM SecurityToGroup definition is inactive.

- **securityToGroup/securityModel**

  An identifier that uniquely identifies a securityModel of the Security Subsystem within the SNMP Management Architecture.

`SNMPv1`	Security model SNMP v1.
`SNMPv2`	Security model SNMP v2.
`USM`	Security model USM.

- **securityToGroup/securityName**

  The securityName for the principal, represented in a Security Model independent format, which is mapped by this entry to a groupName.

- **securityToGroup/groupName**

  The name of the group to which this entry (e.g., the combination of securityModel and securityName) belongs.

Examples are found in Section 4.20.5 on page 79.

4.20.3.3 Element: Accesses

The element `accesses` contains the following parameters:

Configuration format:

```
<accesses>
  <access active="data">
    <groupName>data</groupName>
    <securityModel>data</securityModel>
    <securityLevel>data</securityLevel>
    <contextPrefix>data</contextPrefix>
    <contextMatch>data</contextMatch>
    <readViewName>data</readViewName>
    <writeViewName>data</writeViewName>
    <notifyViewName>data</notifyViewName>
  </access>
</accesses>
```

The parameters contain the following data:

- **active**

  This attribute indicates if the VACM Access definition is active or not.

`yes`	The VACM Access definition is active.
`no`	The VACM Access definition is inactive.

- **groupName**

  The name of the group.

- **securityModel**

  An identifier that uniquely identifies a securityModel of the Security Subsystem within the SNMP Management Architecture.

`SNMPv1`	Security model SNMP v1.
`SNMPv2`	Security model SNMP v2.
`USM`	Security model USM.

- **securityLevel**

  A Level of Security at which SNMP messages can be sent or with which operations are being processed; in particular, one of:

`noAuthNoPriv`	No Authentication and no Privacy used.
`authNoPriv`	Authentication used, but no Privacy.
`authPriv`	Authentication and Privacy used.

- **`contextPrefix`**

  In order to gain the access rights allowed by this conceptual row, a contextName must match exactly (if the value of vacmAccessContextMatch is 'exact') or partially (if the value of vacmAccessContextMatch is 'prefix') to the value of the instance of this object.

- **`contextMatch`**

  If the value of this object is *exact*, then all rows where the contextName exactly matches vacmAccessContextPrefix are selected.

  If the value of this object is *prefix*, then all rows where the contextName whose starting octets exactly match vacmAccessContextPrefix are selected. This allows for a simple form of wildcarding.

`exact`	Exact match of prefix and contextName.
`prefix`	Only match to the prefix.

- **`readViewName`**

  The value of an instance of this object identifies the MIB view of the SNMP context to which this conceptual row authorizes read access.

- **`writeViewName`**

  The value of an instance of this object identifies the MIB view of the SNMP context to which this conceptual row authorizes write access.

- **`notifyViewName`**

  The value of an instance of this object identifies the MIB view of the SNMP context to which this conceptual row authorizes access for notifications.

Examples are found in Section 4.20.5 on page 79.

4.20.3.4 Element: Views

The element `views` contains the following parameters:

`Configuration format:`

```
<views>
  <view active="data">
    <name>data</name>
```

```
    <subtree>data</subtree>
    <mask>data</mask>
    <type>data</type>
  </view>
</views>
```

The parameters contain the following data:

- **active**

  This attribute indicates if the VACM View definition is active or not.

yes	The VACM View definition is active.
no	The VACM View definition is inactive.

- **name**

  The human readable name for a family of view subtrees.

- **subtree**

  The MIB subtree which when combined with the corresponding instance of vacmViewTreeFamilyMask defines a family of view subtrees.

- **mask**

  The bit mask which, in combination with the corresponding instance of vacmViewTreeFamilySubtree, defines a family of view subtrees.

- **type**

  Indicates whether the corresponding instances of vacmViewTreeFamilySubtree and vacmViewTreeFamilyMask define a family of view subtrees which is included in or excluded from the MIB view.

included	Included in the MIB view.
excluded	Excluded from the MIB view.

Examples are found in Section 4.20.5 on page 79.

4.20.4 Configuration Activation

The VACM configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

4.20.5 Examples

4.20.5.1 Example securityToGroup

The following example shows a group being specified for the security name `v1v2ReadOnlySecName`.

```
<securityToGroup active="yes">
  <securityModel>SNMPv2</securityModel>
  <securityName>v1v2ReadOnlySecName</securityName>
  <groupName>v1v2ReadOnlyGroup</groupName>
</securityToGroup>
```

4.20.5.2 Example: access

The following example shows an access being defined for security name `v1v2ReadOnlyGroup`.

```
<accesses>
  <access active="yes">
    <groupName>v1v2ReadOnlyGroup</groupName>
    <securityModel>SNMPv2</securityModel>
    <securityLevel>noAuthNoPriv</securityLevel>
    <contextPrefix/>
    <contextMatch>exact</contextMatch>
    <readViewName>fullReadView</readViewName>
    <writeViewName>noWriteView</writeViewName>
    <notifyViewName>fullNotifyView</notifyViewName>
  </access>
</accesses>
```

4.20.5.3 Example: view

The following example shows a view named `fullReadView` being defined.

```
<views>
  <view active="yes">
    <name>fullReadView</name>
    <subtree>1.3</subtree>
    <mask/>
    <type>included</type>
  </view>
</views>
```

5 Administration

5.1 Version

5.1.1 Introduction

This section describes how to find the software versions of ESA and SSM version that is installed on a system.

The software versions are also accessible using SNMP. See section Section 4.8 on page 45.

5.1.2 ESA Version

The ESA version is found by executing the following commands.

```
# cd <esa basedir>/lib
```

```
# java -jar esa.jar
```

The following is the output format:

```
Name=Ericsson SNMP Agent
```

```
NameAbbrev=ESA
```

```
Version=X.X
```

```
Build=X.X.<version>.<build>
```

```
BuildDate=<date> <time>
```

5.1.3 SSM Version

The SSM version is found by executing the following commands.

```
# cd <ssm basedir>/bin
```

```
# ssmagent -v
```

The following is an output example:

```
4.0.1-155 (Nov 19 2014)
```

5.2 Logging

5.2.1 Introduction

This section describes the log files produced by the ESA, how to configure the log content, where to find the logs and some words about the data found within them. The logging component log4j is used for all logs.

The following log files are used in the ESA:

- Master Agent log

  This log file contains log entries related to the Master Agent operations.

  Appender `masterAgentLogFileAppender` is defined.

- FM Agent log

  This log file contains log entries related to the FM Agent operations.

  Appender `fmAgentLogFileAppender` is defined.

- PM Agent log

  This log file contains log entries related to the PM Agent operations.

  Appender `pmAgentLogFileAppender` is defined.

- Alarms log

  This log file contains every alarm and event that has been triggered in the FM Agent, which means all ‘‘alarm raise’’, ‘‘alarm clear’’ and ‘‘event’’ are logged.

  Appender `alarmLogFileAppender` is defined.

- Nontranslated alarms log

  This log file contains every trap and notification that has gone through the alarm translation process in the FM Agent *not being successfully* matched to a condition. The complete trap content is logged in a log file named `<ipaddress>.log`, where `<ipaddress>` is the IP address of the alarming source, and stored in sub directory `/nontranslated` of the main log directory.

  Appender `nonTranslatedLogFileAppender` is defined.

- Translated alarms log

  This log file contains every trap and notification that has gone through the alarm translation process in the FM Agent *being successfully* matched to a condition. The complete trap content is logged in a log file named

<*ipaddress*>.log, where <ipaddress> is the IP address of the alarming source, and stored in sub directory /translated of the main log directory.

Appender translatedLogFileAppender is defined.

Please note that this log is the only log file not active by default. To activate it see instructions in Section 5.2.3.2 on page 84.

- SSM logs

  The optional component SSM has its own log directory with own log files.

The log file directories are defined in log file configuration files.

5.2.2 Configuration Files

The following configuration files are used for log file administration.

- log4jFmAgent.xml

  Holds the configurations for the following log files.

  - FmAgent.log
  - alarms.log
  - /nontranslated/<*ip addr*>.log
  - /translated/<*ip addr*>.log

- log4jMasterAgent.xml

  Holds the configuration for the MasterAgent.log.

- log4jPmAgent.xml

  Holds the configuration for the PmAgent.log.

All configuration files are found in directory *{esa basedir}*/conf.

5.2.3 Configuration Format

5.2.3.1 Overview

The log files are configured using the configuration format of log4j. See the log4j DTD, which is found as DTD file log4j.dtd in directory *{esa basedir}*/conf, and the log4j documentation at Reference [4].

Normally the log4j configurations are not changed, but there are three parameter that, from time to time, might need modification.

- Log Activation/Deactivation
- Log Level
- Log Directory

5.2.3.2 Log Activation/Deactivation

The activation and deactivation of the log file creation is managed using values `all` and `off` for the `value` attribute to the `logger` element.

The following values are supported.

- `all` (recommended level)

  Logging is active.

- `off`

  Logging is inactive.

All logging is by default active, except the translated log which by default is inactive.

The following is an example configuration.

Configuration example:

```
...
 <logger name="alarmRaiseLog" additivity="false">
  <level value="all" />
  <appender-ref ref="alarmLogFileAppender" />
 </logger>
...
 <logger name="translatedLog" additivity="false">
  <level value="OFF" />
  <appender-ref ref="translatedLogFileAppender" />
 </logger>
...
```

5.2.3.3 Log Level

The log level can be changed to higher or lower level of log details. The log level configuration is done in the log files described in Section 5.2.2 on page 83.

The following log levels are supported.

- `TRACE`
- `DEBUG`
- `INFO` (recommended level)

- WARN
- ERROR
- FATAL

Default value is INFO, but in case the ESA is not working well, the level DEBUG could be useful. If the ESA is working well and the log entries are found too detailed the level WARN or ERROR can be used. It is however recommended to keep level INFO for normal operation.

The following is an example of a log level configuration.

Configuration example:

```
...
<category name="com.ericsson.esa">
  <priority value="INFO" />
</category>
...
```

5.2.3.4 Log Directory

The target directories where the log files are stored can be modified. The directory configuration is done in the log files described in Section 5.2.2 on page 83.

The following list presents the *default* log directories. The directories are by default the same for all log files, but they can be changed individually.

- **Linux/Unix**

  /var/log/esa

- **Windows**

  C:\Program Files\Ericsson\Ericsson SNMP Agent\log

The default target directory for the PM Agent 3GPP XML output files is the following.

- **Linux/Unix**

  /var/log/esa/pm3gppXml

- **Windows**

  C:\Program Files\Ericsson\Ericsson SNMP Agent\log\pm3gppXml

The following is an example of a log directory configuration.

Configuration example:

```
...
<appender name="fileAppender"
    class="org.apache.log4j.RollingFileAppender">
  <param name="File"
      value="/var/log/esa/FmAgent.log"/>
...
```

5.2.4 Configuration Activation

The logging configurations are read by ESA during runtime, which means that the ESA does **not** need to be restarted to make the changes take effect.

5.2.5 Default Warning Messages

The following warning messages are by default found in the Master Agent log file `MasterAgent.log` as well as in the FM Agent log file `FmAgent.log`. There is no need to take any action on these warning messages. They are completely normal.

- **First Master Agent start**

  At the very first start of the Master Agent, it does not find the file `esama.bc` in directory *{esa basedir}*`/bin`. The following is logged in the Master Agent log.

  Example:

  ```
  WARN <timestamp> [main] org.snmp4j.agent.cfg.EngineB
  ootsCounterFile - Could not find boot counter file:
  <path>/esama.bc
  ```

- **First FM Agent start**

  At the very first start of the FM Agent, it does not find the file `esafma.bc` in directory *{esa basedir}*`/bin`. The following is logged in the FM Agent log.

  Example:

  ```
  WARN <timestamp> [main] org.snmp4j.agent.cfg.EngineB
  ootsCounterFile - Could not find boot counter file:
  <path>/esafma.bc
  ```

- **AgentX registration**

  At the registration of any AgentX subagent, the Master Agent issues a "AgentX connection warning". This is found specifically for the FM Agent

and PM Agent, which are part of the ESA. The following is logged in the Master Agent log.

Example:

```
WARN <timestamp> [DefaultTCPTransportMapping_0.0.0.0/705
] org.snmp4j.agent.agentx.master.AgentXCommandProcessor
- SysOREntry managed object for context not found,
instead found:  null
```

5.2.6 Logging Alarms

The log file `alarms.log` contains every alarm and event that has been sent from the FM Agent to the Master Agent. Please note that the log shows what was sent from the FM Agent and not what is received in the OSS. The Master Agent might stop the message from being sent due to mismatched SNMPv3 settings between ESA and the OSS or lack of or erroneous trap destination configuration.

Format for message types `Alarm Raise` and `Alarm Clear`.

```
!----- <Message Type> ---------------------------------
Module              : <Module>
Error Code          : <Error Code>
Resource Id         : <Resource Id>
Timestamp First     : <Timestamp First>
Repeated Counter    : <Repeated Counter>
Timestamp Last      : <Timestamp Last>
Model Description   : <Model Description>
Active Description  : <Active Description>
Event Type          : <Event Type>
Probable Cause      : <Probable Cause>
Severity            : <Severity>
Orig Source IP      : <Originating Source IP>
Sequence Number     : <Sequence Number>
-------------------------------------------------------!
...
```

Format for message type `Event`.

```
!----- <Message Type> ---------------------------------
Module              : <Module>
Error Code          : <Error Code>
Resource Id         : <Resource Id>
Timestamp           : <Timestamp>
Model Description   : <Model Description>
Active Description  : <Active Description>
Event Type          : <Event Type>
Probable Cause      : <Probable Cause>
Severity            : <Severity>
```

```
Orig Source IP      : <Originating Source IP>
Sequence Number     : <Sequence Number>
-----------------------------------------------------!
...
```

The rows are interpreted as follows. The detailed descriptions of each parameter is found in Reference [2].

- **Message Type**

  The message type indicates the SNMP message type that was sent; `Alarm Raise`, `Alarm Clear` or `Event`.

- **Module**

  This is the parameter `moduleId`, which is found in the Alarm Definition configuration file.

- **Error Code**

  This is the parameter `errorCode`, which is found in the Alarm Definition configuration file.

- **Resource Id**

  This is the `Resource Identity` of the alarming source. This indicates the alarming object in the system topology configuration file.

- **Timestamp**

  This is the timestamp indicating when the event was sent.

- **Timestamp First**

  This is the timestamp indicating when the alarm raise was initially sent.

- **Repeated Counter**

  This is a counter indicating how many times the alarm has been repeated between the first and the last timestamp.

- **Timestamp Last**

  This is the timestamp indicating when the alarm raise and alarm clear was sent. If the Repeated Counter indicates ‘‘1’’ the first and last timestamps are the same.

- **Model Description**

  This is the parameter `modelDescription`, which is found in the Alarm Definition configuration file.

- **Active Description**

This is the real time description of the alarm. It is given at the trigger time of the alarm, and not taken from a configuration file. If such description is *not* given at trigger time, this row will be the same as the parameter `activeDescription`, which is found in the Alarm Definition configuration file.

- **Event Type**

  This is the parameter `eventType`, which is found in the Alarm Definition configuration file.

- **Probable Cause**

  This is the parameter `probableCause`, which is found in the Alarm Definition configuration file.

- **Severity**

  This is the parameter `severity`, which is found in the Alarm Definition configuration file.

- **Originating Source IP**

  This is the IP address of the node that triggered the alarm. This is taken from the application triggering the trap.

- **Sequence Number**

  This number is increased by 1 for each alarm and event that is sent, which means there is one sequence number for message type ‘‘alarm’’ and one sequence number for message type ‘‘event’’.

Example

The following is an example log file output showing one alarm raise, one alarm clear and one event.

```
!----- Alarm Raise ----------------------------------------
Module              : MYALARM
Error Code          : 10
Resource Id         : 1.1.1.2.3
Timestamp First     : Mon May 05 15:01:00 CEST 2014
Repeated Counter    : 1
Timestamp Last      : Mon May 05 15:01:00 CEST 2014
Model Description   : My test alarm.
Active Description  : My test alarm.
Event Type          : 2
Probable Cause      : 119
Severity            : warning
Orig Source IP      : 19.72.9.21
Sequence Number     : 5
-----------------------------------------------------------!
!----- Event ----------------------------------------------
```

```
Module              : MYEVENT
Error Code          : 20
Resource Id         : 1.1.1.2.8
Timestamp           : Mon May 05 15:02:00 CEST 2014
Model Description   : My test event.
Active Description  : My test event.
Event Type          : 2
Probable Cause      : 119
Severity            : cleared
Orig Source IP      : 19.72.9.21
Sequence Number     : 12
-----------------------------------------------------!
!----- Alarm Clear -----------------------------------
Module              : MYALARM
Error Code          : 10
Resource Id         : 1.1.1.2.3
Timestamp First     : Mon May 05 15:01:00 CEST 2014
Repeated Counter    : 6
Timestamp Last      : Mon May 05 15:31:00 CEST 2014
Model Description   : My test alarm.
Active Description  : My test alarm.
Event Type          : 2
Probable Cause      : 119
Severity            : cleared
Orig Source IP      : 19.72.9.21
Sequence Number     : 5
-----------------------------------------------------!
```

5.2.7 Logging Not Translated Alarms

This log file contains every trap and notification that has gone through the alarm translation process in the FM Agent without being successfully matched to a condition. In other words the log contains traps that are *not translated*.

Format

A single log entry for one SNMP trap/notification being *not translated* looks like the following.

```
!----- <timestamp> -----------------------------------
Trap OID: <trap oid>
1: <oid varbind 1>, <snmp datatype>, <value>
2: <oid varbind 2>, <snmp datatype>, <value>
...
n: <oid varbind n>, <snmp datatype>, <value>
-----------------------------------------------------!
```

Example

This log file is an example derived from the FM Agent showing a few SNMP traps that are not translated.

```
!----- 2015-02-23 18:33:16 -------------------------------
Trap OID: 1.3.6.1.6.3.1.1.5.1
-------------------------------------------------------------!
!----- 2015-02-23 19:41:06 -------------------------------
Trap OID: 1.3.6.1.4.1.1977.0.38
1: 1.3.6.1.4.1.1977.1.1.2.1.0, OctetString, d7:9b:f9:1d:83:d7:9c:41
2: 1.3.6.1.4.1.1977.1.1.2.2.0, IpAddress, 136.225.39.17
3: 1.3.6.1.4.1.1977.1.1.2.3.0, OctetString, SE00003345
4: 1.3.6.1.2.1.2.2.1.6.14, OctetString, 00:1f:b5:23:d7:85
-------------------------------------------------------------!
!----- 2015-02-23 20:12:08 -------------------------------
Trap OID: 1.3.6.1.4.1.1977.0.46
1: 1.3.6.1.4.1.1977.30.1.1.3.1, OctetString, c:\temp\testlog.txt
2: 1.3.6.1.4.1.1977.30.3.1.3.1.1.1, OctetString, A test error
3: 1.3.6.1.4.1.1977.30.3.1.4.1.1.1, Integer, 3
4: 1.3.6.1.4.1.1977.30.1.1.14.1, OctetString, Mount connection
-------------------------------------------------------------!
```

5.2.8 Logging Translated Alarms

This log file contains every trap and notification that has gone through the alarm translation process in the FM Agent and being successfully matched to a condition. In other words the log contains traps that are *translated*.

This logging is by default inactive. Translated alarms will end up as ESA SNMP alarms being sent to the OSS. If logging of translated alarms is needed it is activated in the log configuration, see Section 5.2.3.2 on page 84.

Format

A single log entry for one SNMP trap/notification being *translated* looks like the following.

```
!----- <timestamp> ---------------------------------------
Trap OID: <trap oid>
Sender IP: <sender IP address>
1: <oid varbind 1>, <snmp datatype>, <value>
2: <oid varbind 2>, <snmp datatype>, <value>
...
n: <oid varbind n>, <snmp datatype>, <value>
*** <operation>
Type               : <messageType>
Module             : <moduleId
Error Code         : <errorCode
Resource Id        : <resourceId>
Orig Source IP     : <origSrcIp>
Active Description : <activeDescr>

-------------------------------------------------------------!
```

The format above shows the SNMP alarm and content being received and also what operation is taken; Translation, Suppress or Transparent. One alarm could also trigger multiple operations.

Example

This log file is an example derived from the FM Agent showing a SNMP trap that is translated.

```
!----- 2015-02-23 18:33:16 ------------------------------
Trap OID: 1.3.6.1.4.1.193.3
Sender IP: 127.0.0.1
1: 1.3.6.1.4.1.193.1.1.1, OctetString, Test string
2: 1.3.6.1.4.1.193.1.1.2, OID, 1.1.1.2.3
3: 1.3.6.1.4.1.193.1.1.3, UnsignedInteger32, 1000
4: 1.3.6.1.4.1.193.1.1.4, Gauge32, 2000
5: 1.3.6.1.4.1.193.1.1.5, Counter32, 4294967295
6: 1.3.6.1.4.1.193.1.1.6, Counter64, 1844674407370955
7: 1.3.6.1.4.1.193.1.1.7, TimeTicks, 0:20:34.56
8: 1.3.6.1.4.1.193.1.1.8, IpAddress, 10.0.0.3
9: 1.3.6.1.4.1.193.1.1.9, IpAddress, 0:0:0:0:0:0:0:1
10: 1.3.6.1.4.1.193.1.1.10, OctetString, String with newline
Text on new row.
11: 1.3.6.1.4.1.193.1.1.11, Integer32, 1234

*** Translation
Type               : Alarm raise
Module             : HARDWARE
Error Code         : 8001
Resource Id        : 1.1.1.2.3
Orig Source IP     : 10.0.0.3
Active Description : Test hw alarm. Fan speed: 1000.
-------------------------------------------------------!
```

5.2.9 Logging SSM

The log files for the SSM component are found in separate directories from the ESA itself.

- ***{ssm basedir}*/log/**

  This directory contains the log file `ssmagent.log`.

See the Netcool SSM documentation for more information about the SSM log files.

5.3 Backup and Restore

5.3.1 Overview

The ESA comes with a backup command that simplifies ESA backup of all configuration files by using a single command. The command is copying all the ESA configuration files from the specified configuration directories to a target directory given as an argument to the command. The same command can be used to restore a previous backup.

The command also comes with the option to include all the log files together with the configuration files in the backup operation. This option is useful when executing a backup in order to send all configuration files and log files to ESA support for support and trouble shooting purposes. The option is not valid in the restore operation.

Please note that the SSM, if installed, requires a separate backup procedure, since the SSM configuration is not backed up with the ESA backup command.

All backup and restore procedures for both the ESA and the SSM are described in the following chapters.

5.3.2 ESA Backup and Restore

5.3.2.1 Format

The command `esabackup` is used for both backup and restore of all the ESA configuration files into and from a backup repository.

`Command format:`

`esabackup` *`<action>`* `[<options>]` *`<target dir>`*`|`*`<source dir>`*

The command attributes are the following:

- **`action`**

  `-b` Execute a backup operation.

  `-r` Execute a restore operation.

- **`options`**

  `-l` Also include the log files in the backup operation.

  Note: This option is not valid for the restore operation.

- **`target dir`**

This is the target directory where the backup operation will store the ESA files. The backup command creates a new directory, within the target directory, which is named with date and time of executing the backup command according to the following format.

/<target dir>/<YYYYMMDD_hhmmss>/

- **source dir**

  This is the source directory from where the restore operation will get the ESA files. The full path shall be given, which means including the directory name created using date and time as reference.

5.3.2.2 Operation

Execute the following steps to perform a backup.

1. The backup is executed no matter if the ESA is running or not. There is therefore no need to stop the ESA prior to backup.
2. Perform the backup operation.

   # **esabackup -b** ***<target dir>***

   Note: Add option `-l` to also include the log files in the backup.

Caution!

Keep in mind that using the restore operation will overwrite the existing ESA configuration files!

Execute the following steps to perform a restore.

1. Stop the ESA.
2. Perform the restore operation.

   # **esabackup -r** ***<source dir>***

3. Start the ESA.

5.3.2.3 Example

The following is a backup example.

□ **Example**

Backup of the ESA configuration that was done September 21st 2014 at time stamp 12:34:56 is done by executing the following command line:

```
# esabackup -b /var/esa/backup
```

The files are stored in a directory looking like the following:

```
/var/esa/backup/20140921_123456/
```

Restore of the ESA configuration that was done September 21st 2014 at time stamp 12:34:56 is done by executing the following command line:

```
# esabackup -r /var/esa/backup/20140921_123456/
```

5.3.3 SSM Backup and Restore

Perform the following steps to perform a backup of SSM configuration:

1. Make sure that the SSM is started.
2. Save the SSM configuration and state by using the SSM Console.

   ```
   # {ssm basedir}/bin/ssmcons -p 7161

   Agent> config save

   Agent> quit
   ```

   Note: The SSM configuration is not found in this document. The port number is by default defined to 7161 during the SSM installation. This port number can be changed to a number of own choice.

3. Backup the following directory.

   {ssm basedir}/config

Perform the following steps to perform a restore of SSM configuration:

1. Start the SSM.
2. Copy the SSM configuration files from the backup directory to directory *{ssm basedir}*/config.
3. Load the SSM configuration and state by using the SSM Console.

   ```
   # {ssm basedir}/bin/ssmcons -p 7161

   Agent> config execute agent.state

   Agent> warmboot

   Agent> quit
   ```

5.4 Command Line Interface

The CLI commands provided by the ESA are described in the documents and chapters that are related to the feature and function of the commands, such as backup command is described in this document, FM commands in Reference [2] and PM commands in Reference [3].

Please note that the very first time a CLI command is executed after installed it will show the following printout:

```
testing JVM in /usr ...
```

The printout simply indicates that the command is searching for and testing the java version prior to executing the CLI command. This can take some time on slower machines, which is the reason for having the printout. A user waiting for the command to execute simply needs to know what is happening. This will however only happen the very first time any of the ESA commands are executed by a user. After the first execution the path to the JVM is stored and reused in future command executions.

6 Appendix A: ESA MIBs

The MIBs included in the ESA can be found in the following directories.

- *{esa basedir}*/mibs/

The following MIBs are used/published by the ESA:

- **AGENTPP-GLOBAL-REG**

  The root MIB for Agent PP.

- **AGENTX-MIB**

  The MIB for the SNMP Agent Extensibility Protocol (AgentX). Part of RFC 2742.

- **ALARM-MIB**

  The MIB contains parts of the possible alarms ESA can send, the alarm definitions, and statistics about the alarm state and parts of an active alarm list. See also `itu-alarm-mib.mib`.

- **ERICSSON-ESA-INFORMATION-MIB**

  The MIB contains general information about the ESA and the system where the ESA resides. For example version, install date, system name and more.

- **ERICSSON-ESA-PM-MIB**

  The MIB contains the PM counters collected by the ESA.

- **ERICSSON-ESA-TOP-MIB**

  The MIB contains the definition of the ESA branch under the Ericsson branch.

- **ERICSSON-SNF-ALARM-MIB**

  The MIB contains the definition of alarms used by the ESA.

- **ERICSSON-SNF-EVENT-MIB**

  The MIB containing the definition of the event notification used by ESA.

- **ERICSSON-SNF-MODEL-MIB**

  The MIB containing the definition of the Alarms.

- **ERICSSON-SNF-TOP-MIB**

The MIB contains the definition of the generic SNF branches under the Ericsson branch.

- **ERICSSON-TOP-MIB**

  The MIB contains the definition of the Ericsson branch under the enterprise MIB tree.

- **IANA-ITU-ALARM-TC**

  The MIB contains the definitions of EventType and ProbableCause (according to X.733).

- **INET-ADDRESS-MIB**

  The MIB is a standard MIB holding textual conventions that are imported by the ESA MIBs.

- **ITU-ALARM-MIB**

  The MIB contains parts of the possible alarms ESA can send, the alarm definitions, and statistics about the alarm state and parts of an active alarm list. See also `alarm-mib.mib`.

- **ITU-ALARM-TC**

  The MIB contains the severity definitions (according to X.733).

- **NOTIFICATION-LOG-MIB**

  The MIB for logging SNMP Notifications, that is Traps and Informs.

- **SNMP-COMMUNITY-MIB**

  The MIB defines objects to help support coexistence between SNMPv1, SNMPv2c, and SNMPv3.

- **SNMP-FRAMEWORK-MIB**

  The MIB is a standard MIB holding textual conventions that are imported by the ESA MIBs.

- **SNMP-MPD-MIB**

  The MIB for Message Processing and Dispatching. Part of RFC 3412.

- **SNMP-NOTIFICATION-MIB**

  The MIB defines MIB objects which provide mechanisms to remotely configure the parameters used by a SNMP entity for the generation of notifications. Part of RFC 2273.

- **SNMP-PROXY-MIB**

This MIB module defines MIB objects which provide mechanisms to remotely configure the parameters used by a proxy forwarding application. Part of 2273.

- **SNMP-TARGET-MIB**

  This MIB module defines MIB objects which provide mechanisms to remotely configure the parameters used by a SNMP entity for the generation of SNMP messages. Part of 2273.

- **SNMP-USER-BASED-SM-MIB**

  The management information definitions for the SNMP User-based Security Model. Part of RFC 2274.

- **SNMP-VIEW-BASED-ACM-MIB**

  The management information definitions for the View-based Access Control Model for SNMP. Part of RFC 2575.

- **SNMP4J-AGENT-REG**

  This MIB specification is the global registration module for SNMP4J-Agent related MIBs.

- **SNMP4J-CONFIG-MIB**

  This MIB module defines managed objects for the configuration of a SNMP4J agent.

- **SNMP4J-LOG-MIB**

  The SNMP4J-LOG-MIB defines objects for the management of the logging system of a SNMP4J-Agent enabled system.

- **SNMP4J-PROXY-MIB**

  This MIB module defines MIB objects which provide mechanisms to remotely configure the parameters for subtree proxy applications.

- **SNMPv2-MIB**

  The MIB module for SNMP entities.

- **TRANSPORT-ADDRESS-MIB**

  This MIB module provides commonly used transport address definitions. Part of 3419.

7 Appendix B: SNMP Configuration

7.1 Introduction

This appendix gives an overview of how SNMPv2 and SNMPv3 can be setup using the ESA. The concepts in this section are generic for SNMP, but the example configurations are in ESA format.

Please note that configuration descriptions for each configuration file presented in this appendix is in more detail described in the sub sections of Section 4 on page 15.

7.2 Overview

The appendix shows how to setup the following use cases:

- SNMP get for SNMPv2 (valid also for SNMP set)
- SNMP trap for SNMPv2
- SNMP get for SNMPv3 (valid also for SNMP set)
- SNMP trap for SNMPv3

7.3 Use Case: SNMP Get for SNMPv2

This use case is about sending an "SNMP get" request from an OSS to the ESA in SNMPv2 mode.

The following picture shows an overview of how the operation is executed by the ESA using the SNMP configuration settings. Please note that the execution is the same for a "SNMP set" operation.

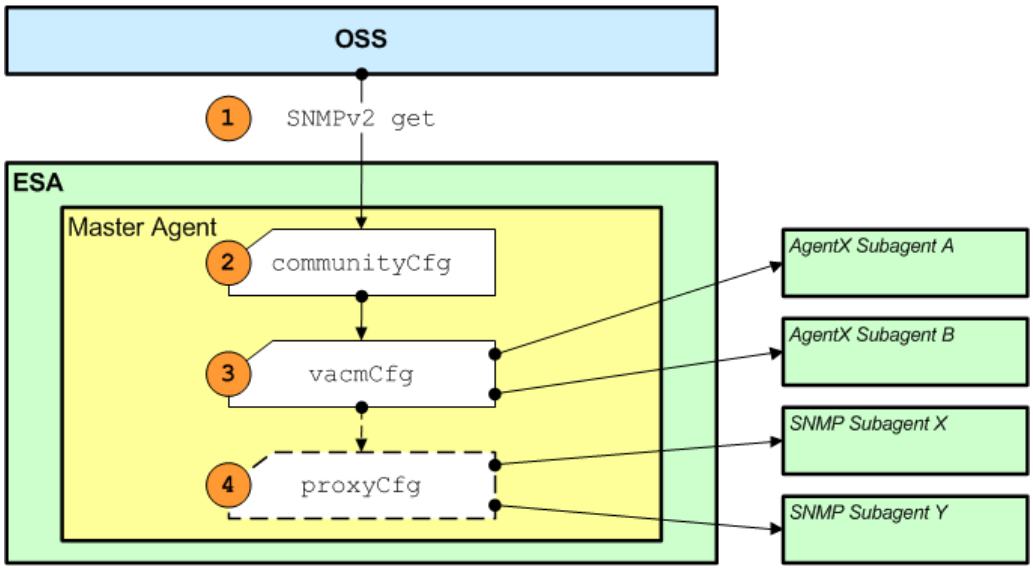

Figure 10 Execution of SNMP get operation for SNMPv2 configuration.

The following steps are taken:

1 The SNMPv2 get message is sent from the OSS. It contains the following data:

```
Community String: ESA-PC
Target Oid: .1.3.6.1.2.1.1.1 (= mib-2.system.sysDescr)
```

2 The community string in the SNMP get message is matched with the community strings configured in the ESA.

The following example is setup in the ESA.

```
<communityCfg>
  <community active="yes">
    <communityString>ESA-PC</communityString>
    <securityName>v1v2ReadOnlySecName</securityName>
  </community>
  <community active="yes">
    <communityString>ESA-PE</communityString>
    <securityName>v1v2ReadWriteSecName</securityName>
  </community>
</communityCfg>
```

The community string configuration shows that the community string in the SNMP get message is valid and access is granted.

Also, it can be seen that the community string ESA-PC is mapped to the Security Name "v1v2ReadOnlySecName".

3 The SNMP get requires access to the OID being targeted. This is controlled by the VACM configuration.

The following example is setup in the ESA.

```
<vacmCfg>
  <securityToGroups>
    ...
    <securityToGroup active="yes">
      <securityModel>SNMPv2</securityModel>
      <securityName>v1v2ReadOnlySecName</securityName>
      <groupName>v1v2ReadOnlyGroup</groupName>
    </securityToGroup>
    ...
  </securityToGroups>
  <accesses>
    ...
    <access active="yes">
      <groupName>v1v2ReadOnlyGroup</groupName>
      <securityModel>SNMPv2</securityModel>
      <securityLevel>noAuthNoPriv</securityLevel>
      <contextPrefix/>
      <contextMatch>exact</contextMatch>
      <readViewName>fullReadView</readViewName>
      <writeViewName>noWriteView</writeViewName>
      <notifyViewName>fullNotifyView</notifyViewName>
    </access>
    ...
  </accesses>
  <views>
    ...
    <view active="yes">
      <name>fullReadView</name>
      <subtree>1.3</subtree>
      <mask/>
      <type>included</type>
    </view>
    ...
  </views>
```

The VACM configuration above shows the following:

- In securityToGroups; The Security Name "v1v2ReadOnlySecName" is an "SNMPv2" security model.
- In securityToGroups; The Security Name "v1v2ReadOnlySecName" belongs to groups "v1v2ReadOnlyGroup".
- In accesses; The group "v1v2ReadOnlyGroup" with security model "SNMPv2" has the Security Level "noAuthNoPriv", which indicates use of SNMPv2 (no use of USM and thus not SNMPv3).
- In accesses; The readViewName, which corresponds to the SNMP get operation, points at the view named "fullReadView".

– In views; The view "fullReadView" shows reading of OIDs in OID branch "1.3" is allowed, which means the OID in the SNMP message being ".1.3.6.1.2.1.1.1" is allowed to be read.

At this point access is granted to the requested OID.

If the OID points at an AgentX subagent the ESA MA will route the request to the registered agent and thus bypass the SNMP proxy configuration.

If the OID points at a SNMP subagent, please go to the next step.

4 The ESA MA will now route the request to the correct SNMP subagent holding the requested OID.

The following example is setup in the ESA.

```
<proxyCfg>
  ...
  <proxyDefinition active="yes">
    <proxy proxyType="readAndWrite">
      <subtree>1.3.6.1.2.1.1</subtree>
    </proxy>
    <target snmpVersion="v2c">
      <ip>127.0.0.1</ip>
      <port>7161</port>
      <securityName>private</securityName>
      <securityLevel>noAuthNoPriv</securityLevel>
    </target>
  </proxyDefinition>
  ...
</proxyCfg>
```

The Proxy configuration above shows the following:

– The configuration is valid for all OIDs within branch "1.3.6.1.2.1.1", which matches the OID ".1.3.6.1.2.1.1.1" found in the SNMP get message.

– The target is a SNMP agent using SNMPv2c and is reached at localhost and port 7161.

– The target using SNMPv2 requires community string only, which is represented by Security Name "private".

– The target is not using SNMPv3 and thus indicated with Security Level "noAuthNoPriv".

– Please note that a SNMPv2 targets do not have any entries in the USM configuration.

7.4 Use Case: SNMP Trap for SNMPv2

This use case is about sending an "SNMP notification" (a trap) from the ESA to the OSS in SNMPv2 mode.

The following picture shows an overview of how the operation is executed by the ESA using the SNMP configuration settings.

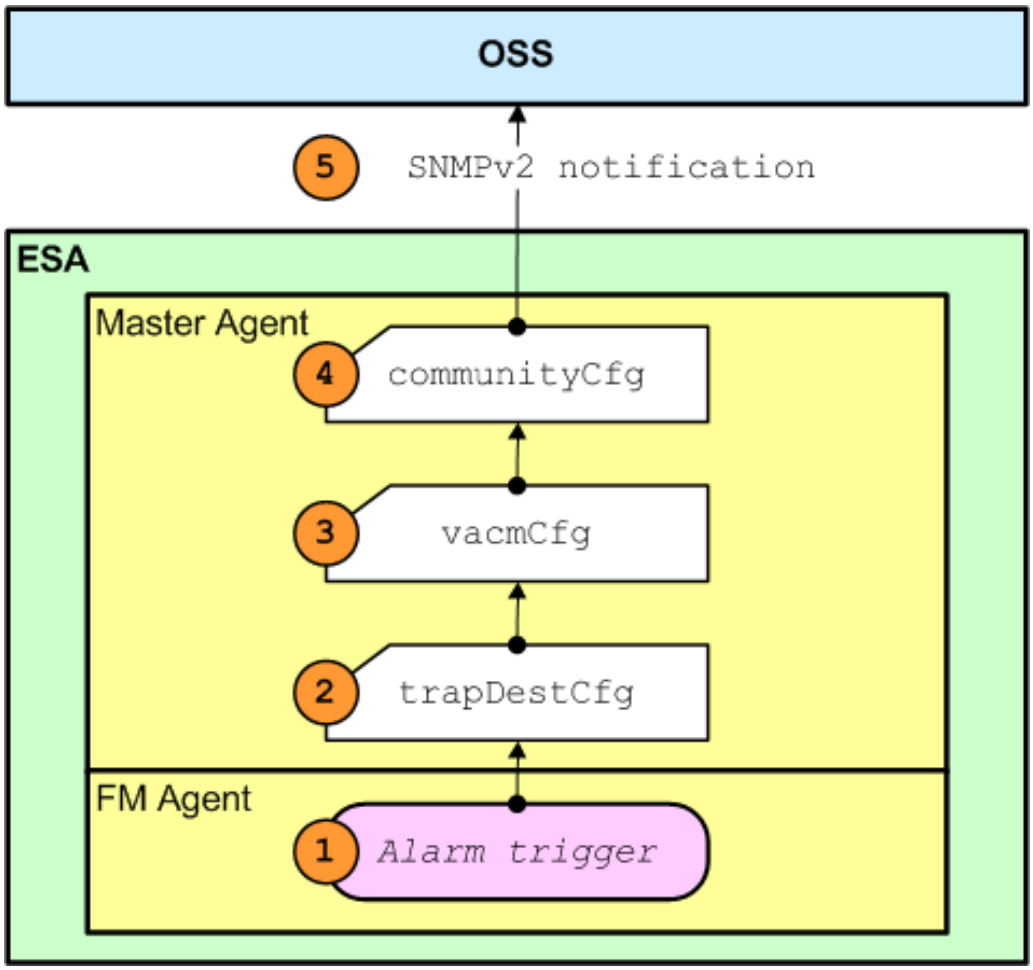

Figure 11 Execution of SNMP trap operation for SNMPv2 configuration.

The following steps are taken:

1 a SNMPv2 notification with OID "1.3.6.1.4.1.193.110.2.10.2.0.1", which is the ESA alarm "alarm raise", is triggered in the ESA FMA. The trap is created by the FMA and sent to the MA.

2 The trap destination is defined according to the following:

```
<trapDestCfg>
  <managerDefinition snmpVersion="v2c" active="yes">
    <ip>100.12.34.56</ip>
```

```
    <port>162</port>
    <securityName>v1v2ReadWriteSecName</securityName>
    <securityLevel>noAuthNoPriv</securityLevel>
  </managerDefinition>
</trapDestCfg>
```

The trap destination configuration above shows the following:

- In securityName; Security name "v1v2ReadWriteSecName" is to be used.
- In securityLevel; Security level is set to "noAuthNoPriv", which indicates use of SNMPv2 (no use of USM and thus not SNMPv3).

3 The SNMP notification must be authorized. This is controlled by the VACM configuration.

The following example is setup in the ESA.

```
<vacmCfg>
  <securityToGroups>
    ...
    <securityToGroup active="yes">
      <securityModel>SNMPv2</securityModel>
      <securityName>v1v2ReadWriteSecName</securityName>
      <groupName>v1v2ReadWriteGroup</groupName>
    </securityToGroup>
    ...
  </securityToGroups>
  <accesses>
    ...
    <access active="yes">
      <groupName>v1v2ReadWriteGroup</groupName>
      <securityModel>SNMPv2</securityModel>
      <securityLevel>noAuthNoPriv</securityLevel>
      <contextPrefix/>
      <contextMatch>exact</contextMatch>
      <readViewName>fullReadView</readViewName>
      <writeViewName>fullWriteView</writeViewName>
      <notifyViewName>fullNotifyView</notifyViewName>
    </access>
    ...
  </accesses>
  <views>
    ...
    <view active="yes">
      <name>fullNotifyView</name>
      <subtree>1.3</subtree>
      <mask/>
      <type>included</type>
    </view>
    ...
```

```
</views>
```

The VACM configuration above shows the following:

- In securityToGroups; The Security Name "v1v2ReadWriteSecName" is an "SNMPv2" security model.
- In securityToGroups; The Security Name "v1v2ReadWriteSecName" belongs to groups "v1v2ReadWriteGroup".
- In accesses; The group "v1v2ReadWriteGroup" with security model "SNMPv2" has the Security Level "noAuthNoPriv", which indicates use of SNMPv2 (no use of USM and thus not SNMPv3).
- In accesses; The notifyViewName, which corresponds to the SNMP notification operation, points at the view named "fullNotifyView".
- In views; The view "fullNotifyView" shows that sending traps and notifications with OIDs within the OID branch "1.3" are allowed, which means the trap with OID "1.3.6.1.4.1.193.110.2.10.2.0.1" is allowed to be sent.

4 The ESA MA sends the SNMP notification to the trap destination at IP:PORT = 100.12.34.56:162.

7.5 Use Case: SNMP Get for SNMPv3

This use case is about sending an "SNMP get" request from an OSS to the ESA in SNMPv3 mode.

The following picture shows an overview of how the operation is executed by the ESA using the SNMP configuration settings. Please note that the execution is the same for an "SNMP set" operation.

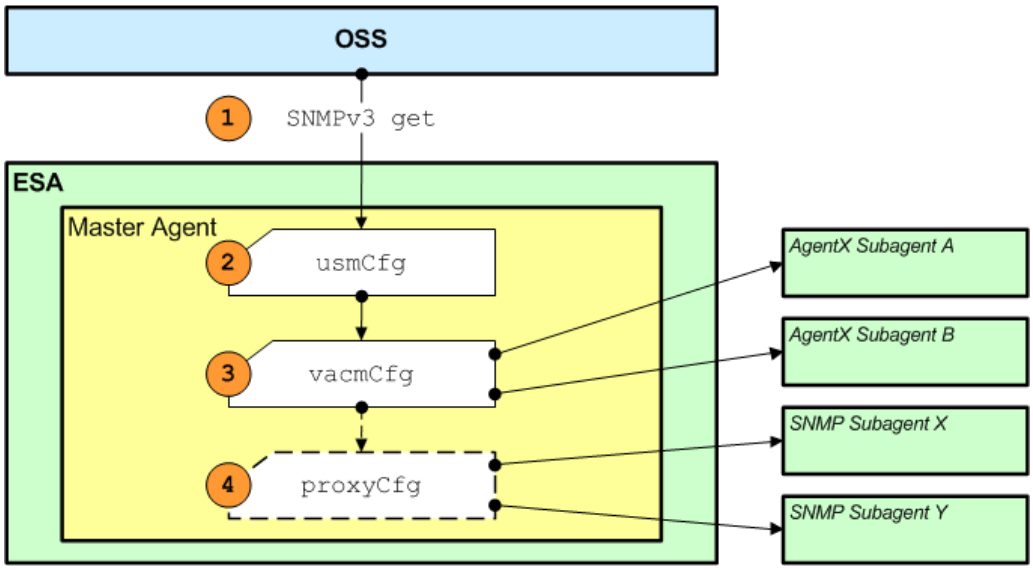

Figure 12 Execution of SNMP get operation for SNMPv3 configuration.

The procedure is the same as for SNMPv2c, but the second step about USM.

1 See Section 7.3 on page 101.

2 The following USM configuration is defined.

```
<usmCfg>
  <usmUser active="yes">
    <securityName>v3ReadWriteSecName</securityName>
    <authProtocol>AuthMD5</authProtocol>
    <authPassword>myAuthPassword</authPassword>
    <privProtocol>PrivAES128</privProtocol>
    <privPassword>myPrivPassword</privPassword>
  </usmUser>
</usmCfg>
```

The USM configuration above shows the following:

- The Security Name "v3ReadWriteSecName" is used and defined.
- Authentication protocol MD5 is used.
- Authentication password is set to "myAuthPassword".
- Privacy encryption AES128 is used.
- Privacy password is set to "myPrivPassword".

This means that if the SNMP get request contains the correct settings and passwords, the request is authenticated and the content is decrypted.

3 The SNMP get requires access to the OID being targeted. This is controlled by the VACM configuration. It is very much the same as for the SNMPv2 case.

```
<vacmCfg>
  <securityToGroups>
    ...
    <securityToGroup active="yes">
      <securityModel>USM</securityModel>
      <securityName>v3ReadWriteSecName</securityName>
      <groupName>v3AuthPrivGroup</groupName>
    </securityToGroup>
    ...
  </securityToGroups>
  <accesses>
    ...
    <access active="yes">
      <groupName>v3AuthPrivGroup</groupName>
      <securityModel>USM</securityModel>
      <securityLevel>authPriv</securityLevel>
      <contextPrefix/>
      <contextMatch>exact</contextMatch>
      <readViewName>fullReadView</readViewName>
      <writeViewName>fullWriteView</writeViewName>
      <notifyViewName>fullNotifyView</notifyViewName>
    </access>
    ...
  </accesses>
  <views>
    ...
    <view active="yes">
      <name>fullReadView</name>
      <subtree>1.3</subtree>
      <mask/>
      <type>included</type>
    </view>
    ...
  </views>
```

The VACM configuration above shows the following:

- In securityToGroups; The Security Name "v3ReadWriteSecName" is an "USM" security model.
- In securityToGroups; The Security Name "v3ReadWriteSecName" belongs to groups "v3AuthPrivGroup".
- In accesses; The group "v3AuthPrivGroup" with security model "USM" has the Security Level "authPriv", which indicates use of SNMPv3.
- In accesses; The readViewName, which corresponds to the SNMP get operation, points at the view named "fullReadView".

– In views; The view "fullReadView" shows reading of OIDs in OID branch "1.3" is allowed, which means the OID in the SNMP message being ".1.3.6.1.2.1.1.1" is allowed to be read.

4 At this point the SNMP get operation enters the same flow of events handling AgentX as in the SNMPv2 case, see Section 7.3 on page 101.

5 At this point the SNMP get operation enters the same flow of events handling SNMP proxies as in the SNMPv2 case, see Section 7.3 on page 101.

7.6 Use Case: SNMP Trap for SNMPv3

This use case is about sending an "SNMP notification" (a trap) from the ESA to the OSS in SNMPv3 mode.

The following picture shows an overview of how the operation is executed by the ESA using the SNMP configuration settings.

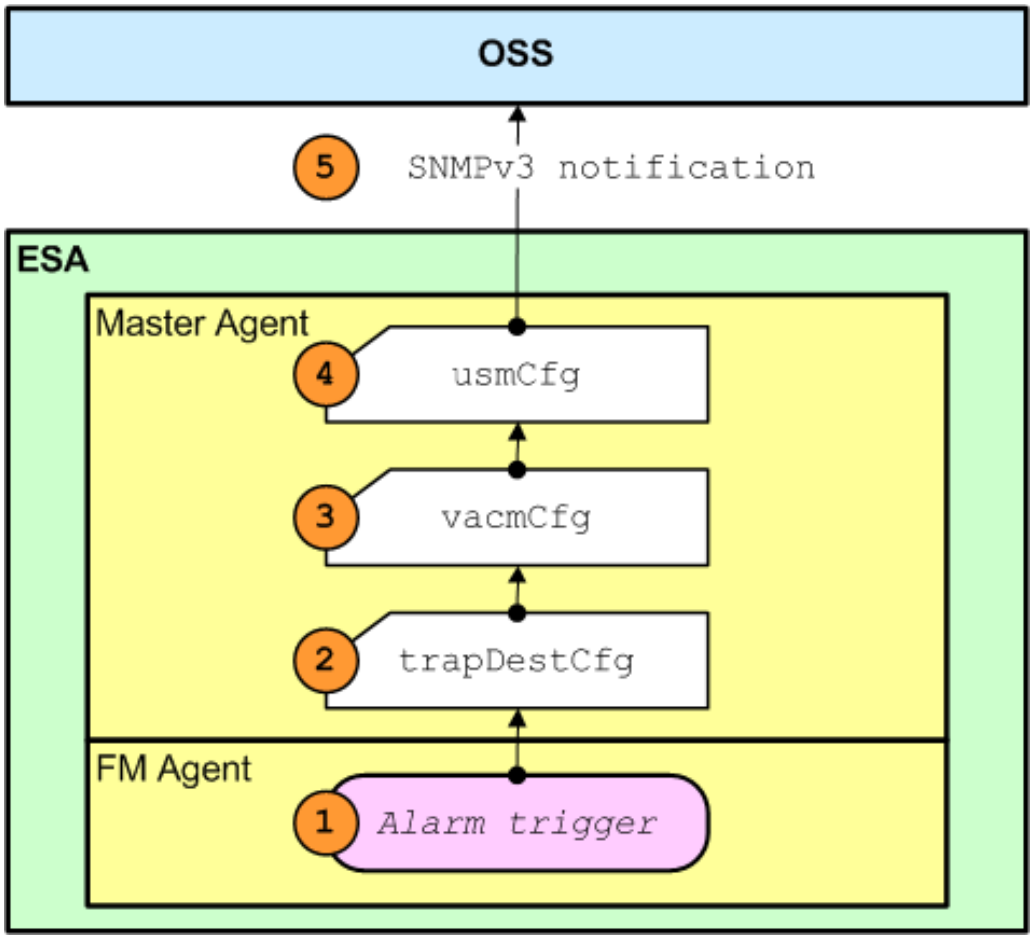

Figure 13 Execution of SNMP trap operation for SNMPv3 configuration.

The procedure is the same as for SNMPv2c, but the fourth step about USM.

1 See Section 7.4 on page 104.

2 The trap destination is defined according to the following:

```
<trapDestCfg>
  <managerDefinition snmpVersion="v3" active="yes">
    <ip>100.12.34.56</ip>
    <port>162</port>
    <securityName>v3ReadWriteSecName</securityName>
    <securityLevel>authPriv</securityLevel>
  </managerDefinition>
</trapDestCfg>
```

The trap destination configuration above shows the following:

– In securityName; Security name "v3ReadWriteSecName" is to be used.

– In securityLevel; Security Level is set to "authPriv", which indicates use of SNMPv3.

3 The VACM configuration is the same as for SNMP get operation in the SNMPv3 case.

4 The following USM configuration is defined for security name "v3ReadWriteSecName".

```
<usmCfg>
  <usmUser active="yes">
    <securityName>v3ReadWriteSecName</securityName>
    <authProtocol>AuthMD5</authProtocol>
    <authPassword>myAuthPassword</authPassword>
    <privProtocol>PrivAES128</privProtocol>
    <privPassword>myPrivPassword</privPassword>
  </usmUser>
</usmCfg>
```

The USM configuration above shows the following:

- The Security Name "v3ReadWriteSecName" is used and defined.
- Authentication protocol MD5 is used.
- Authentication password is set to "myAuthPassword".
- Privacy encryption AES128 is used.
- Privacy password is set to "myPrivPassword".

This means that the SNMP notification is sent encrypted and with authentication, which must match the SNMPv3 configuration in the OSS.

Glossary

Glossary
ESA Glossary of Terms and Acronyms,
0033-CSH 109 532

Reference List

ESA Documentation

[1] *ESA Library Overview*
DIRECTIONS FOR USE, 1/1553-CSH 109 532

[2] *ESA Fault Management*
SYSTEM ADMINISTRATION GUIDE, 2/1543-CSH 109 532

[3] *ESA Performance Management*
SYSTEM ADMINISTRATION GUIDE, 3/1543-CSH 109 532

Web references

[4] *log4j documentation*, http://logging.apache.org/log4j/1.2/

[5] *akka documentation*, http://akka.io/

[6] *oracle documentation*, http://docs.oracle.com/