

Log Management Framework Description

Log Management Framework

FUNCTION DESCRIPTION

Copyright

© Ericsson AB 2017, 2019. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Log Management Framework Overview | 3 |
| 2.1 | Basic Concepts | 3 |
| 3 | Managed Object Model | 7 |
| 4 | Configuration Management | 9 |
| 5 | Log Stream Registration | 11 |
| 5.1 | Log Stream Registration Process | 11 |
| 5.2 | Requirements on Log Stream Owners | 12 |
| 5.3 | Registering SAFLOG Log Streams | 16 |





1 Introduction

This document describes the process of registering log streams with the Log Management (LogM) framework provided by Core Middleware (MW).





2 Log Management Framework Overview

Core MW LogM provides the centralized function to manage log streams. In the Component Based Architecture (CBA) environment, LogM is integrated with the Core MW and Linux Distribution Extensions (LDE) products.

The following functions are included in Core MW LogM:

- Core MW LogM facilitates the registration of log streams in the CBA environment.
- Core MW LogM supports the centralized management by the Northbound Interface (NBI) of registered log streams. The Core MW LogM fragment is derived from the Ericsson Common Information Model (ECIM) LOGM fragment and describes the management interface of the LogM service.
- The operations that can be performed by the Core MW LogM fragment on registered log streams include:
 - Setting severity filter
 - Performing manual export of logs
 - Configuring automatic streaming of log entries towards a log server

CoreMW LogM includes the shared LogM Library used by all services in the Log Management Framework, and the CoreMW Log Controller service that applies the LogM operations to registered SAFLOG log streams.

For more information, refer to [Core MW LogM System Architecture Description](#).

2.1 Basic Concepts

The following concepts are used with Core MW LogM:

Log Stream

All the logs that can be associated to the same family of logs in a cluster (similar generation and storage). For example, all the “messages” logs of a cluster are defined as a log stream.

A log stream is unique per cluster. From a management interface perspective, a log stream is managed by a single instance of the Log class in the Core MW LogM fragment, so the log level of a log stream is managed by a single attribute.

A log stream can be either system or security log stream. A security log stream in CBA must be named using ‘sec_’ prefix.



Log Stream Registration

The initial process that must be performed on a log stream so that it can be managed by the Core MW LogM framework. The operations supported by the Core MW LogM fragment can only be performed on registered log streams.

Log Framework Owner

The component responsible for a log framework. LDE is the log framework owner of syslog and Core MW is the log framework owner of SAFLOG.

Log Stream Owner

The component that initiates the creation of a specific log stream. For example, Core MW is the log stream owner for the SAFLOG “system” stream and LDE is the log stream owner for the “messages” log stream.

Log Stream User

Any component that uses a log stream to collect its own log. The component can be the log stream owner or not. For example, License Manager (LM) is a log stream user for syslog log stream “messages”.

Log Management API

The LogMngt Application Programming Interface (API) is a C-based header file defining the interface between the Log Controller and the shared LogM Library.

Log Controller

A component created for the CBA log framework that handles and applies LogM configurations to the log framework for the registered log streams belonging to it. One Log Controller per Log Framework Owner is required to handle and apply all LogM configurations to all the possible log frameworks in a CBA system. All Log Controllers use the shared LogM Library provided by Core MW LogM and communicate through the LogMgmtController API.

Log Streaming Service

A component created for the CBA log framework that fulfils the automatic log streaming use case and configures the streaming backend to support automatic log streaming over TLS to a remote log server. The Log Streaming Service uses the shared LogM Library provided by Core MW LogM and communicates through the LogMgmtStreaming API.

**LogM Library**

Represents a shared library of common functionalities (export/streaming back end management), an API (LogMgmtController API) to interact with a Log Controller and an API (LogMgmtStreaming API) to interact with the Log Streaming Service. The LogM Library does not contain any functionality specific to a particular log framework. It also implements common functionality for the Information Model Management (IMM) needed for each Log Controller. The LogM Library runs in the process spaces of the Log Controllers and the Log Streaming Service.





3 Managed Object Model

The Core MW LogM fragment is used to define the management interface of the Core MW LogM framework and is derived from the ECIM LOGM model.

The Log Management area is represented in the Managed Object Model (MOM) as follows:

```
ManagedElement
+-SystemFunctions
  +-LogM
    +-Log
      +-LogPushTransfer
```

The Log Management Managed Object Classes (MOCs) are described in Table 1.

Table 1 Core MW LogM Managed Object Class Descriptions

| Managed Object Class | Description |
|----------------------|---|
| LogM | The root of the Log Management model. Specifies the certificates used for automatic log streaming operation towards an external log server. |
| Log | Used to perform export action and to configure the severity filter attribute. One instance is created for each registered log stream. |
| LogPushTransfer | Used to configure or view the state of automatic log streaming operation or to temporarily disable automatic streaming. One instance is created for each registered log stream. |

Note: This is only a short overview of how the model is used with Core MW LogM and is not intended to be the reference information for ECIM and cannot be fully up to date.

For more information about the Core MW LogM fragment, refer to Core MW ECIM Statement of Compliance and Managed Object Model cmw_logm.





4 Configuration Management

Log Management is accessed using NETCONF or the Ericsson Command-Line Interface (ECLI) to manipulate the Management Information Base (MIB).

The following operations can be performed by the user and are described in the following Operating Instructions (OPIs) using the ECLI:

- Set Severity Filter
- Export Log Stream
- Configure Automatic Log Streaming
- View State of Automatic Log Streaming
- Temporarily Disable Automatic Streaming on a Single Log Stream





5 Log Stream Registration

The initial process required before a log stream can be managed by the LogM framework is described as log stream registration.

When log streams are created and opened in a CBA system by the Log Stream Owner, the log streams must be registered for the LogM framework to manage them and handle configuration requests.

This section describes the process of registration and the additional requirements upon Log Stream Owners to be able to register their log streams successfully.

5.1 Log Stream Registration Process

The steps involved in registration flow are the following:

1. LogM Instance models with model type LOGM_R1 are queued by the Model Delivery Framework (MDF) when a Common Component (CC) is installed and delivered immediately if Core MW LogM (the model consumer) is installed. The MDF delivery creates Log Managed Objects (MOs) for the log streams defined in the instance models.
2. Each LogM Library instance is the Object Applier for the Log class and receives a callback for each created MO.
3. The LogM Library instances notifies each running Log Controller in the system of the Log MO instance name via LogMgmtController API callback.

Every Log Controller uses the log stream name and determines if it is responsible to register this log stream by checking if it belongs to the local log environment. For example:

- Core MW Log Controller checks for SAFLOG Configuration streams.
- LDE Log Controller checks against syslog configuration files.

If a Log Controller does not identify the log stream name, then nothing happens. Otherwise it applies the provided severity filter value for the successfully registered log stream.

4. If the Log Controller determines that it is responsible for this log stream, then it registers the log stream using the LogMngt API.
5. The registration is completed.
6. If the Streaming Service for CBA is deployed, it is notified of all registered log streams by a LogM Library instance via LogMgmtStreaming API. This LogM Library instance will create a LogPushTransfer MO instance beneath the registered Log instance. The LogM Library applies the default URI value to

the uri attribute of the new LogPushTransfer as an initial attribute value. The LogM Library instance registers itself as the Object Implementer (OI) for both Log and LogPushTransfer MO instances.

5.2 Requirements on Log Stream Owners

Log Stream Owners have an inventory of log streams, for which they are the application responsible for their creation in a CBA system.

This information must be available when the application starts. The application developer must provide the definitions with the application installation package. This is done by adding static management information to the information model as defined in the Core MW LogM fragment. The definitions are done by creating a Log instance as shown in Figure 1.

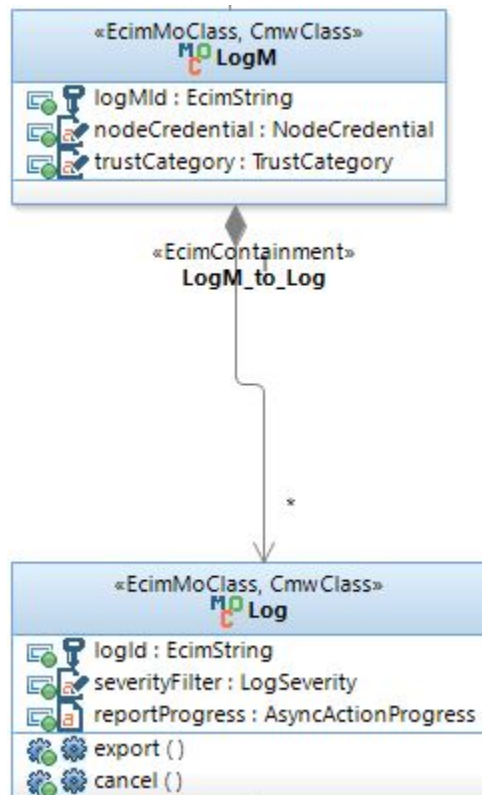


Figure 1 Model for Defining Log Stream of an Application

5.2.1 Generic Requirements on All Log Stream Owners

The generic requirements on Log Stream Owners to register a log stream to be managed by the LogM framework are as follows:

- Design an external Log MO based on Managed Object Model cmw_logm for each log stream.



- Deploy the external Log MO using MDF as dedicated model type LOGM_R1.

For more details on designing the external Log MO, see Section 5.2.3 Design an External Log MO Based on cmw_logm MOM on page 13.

As the Log MO instance is queued and delivered by MDF, there is no installation order dependency for the application towards Core MW LogM. The instances are delivered by MDF once Core MW LogM is installed.

There are also security requirements to enforce NBI access permissions on all log streams.

5.2.2 Framework-Specific Requirements on Log Stream Owners

In addition to the generic requirements, there are additional requirements specific for each log framework.

It is important to ensure that the necessary Log Controller is deployed to support registration. Step 3 and Step 4 involve only one of the deployed Log Controllers successfully identifying the log stream name that is delivered in the Log MO instance, by verifying that the log stream is created in the local log environment that it manages. This identification enables one Log Controller to register that stream and begin handling configuration requests performed on the Core MW LogM fragment.

The following requirements apply to Log Stream Owners:

- The key naming attribute of the Log MO must be identical with the log stream name in the log framework to be successfully registered by a Log Controller.
- Any security log streams shall be named using 'sec_' prefix in CBA.
- For SAFLOG log streams, they must be created as SAFLOG configuration streams to be registered by the Core MW Log Controller. See Section 5.3 Registering SAFLOG Log Streams on page 16.
- To register syslog streams users need to consult the instructions from the relevant log stream controller.
- For any other Log Framework Owners that create a Log Controller, they may have additional requirements on registration that can apply to Log Stream owners.

5.2.3 Design an External Log MO Based on cmw_logm MOM

To design an external MO based on Managed Object Model cmw_logm:

1. An application uses the DX ECIM Tool-chain (ET) with Core MW profile to design the desired external MOs.



2. Include the `cmw_logm` MOM from the Core MW LogM ECIM model in the design project.
3. Reference PM `EcimMocInstance` of the `cmw_logm` MOM under which the designed MO is put. That is, drag and drop the `EcimMocInstance` LogM into the new object diagram.
4. Create the application-specific MO under LogM `EcimMocInstance`.
5. Generate the IMM conformant implementation view models from the UML instance model.
6. Use MDF to deploy the IMM implementation view model using the dedicated model type LOGM_R1.

A new model type LOGM_R1 is introduced by Core MW LogM 4.3. It is recommended that LogM models are delivered through MDF to this model type. For more information, refer to Section Model Delivery Framework in [Core MW System Architecture Description](#).

Deliver Security Rules for Registered Log Streams

It is necessary for log stream owners to design and deliver Security Rule MOs to enforce correct Role Based Access Control as documented in **ECIM SecM Revision C Guidelines** under **Log Management**.

In CBA, the appropriate roles are `SystemAdministrator`, `SystemReadOnly`, and `SystemSecurityAdministrator`.

System log stream owners shall deliver 2 rules applying their Log MO:

- Role=`SystemReadOnly` with R permission
- Role=`SystemAdministrator` with RWX permission

Security log stream owners shall deliver 1 rule applying their Log MO:

- Role=`SystemSecurityAdministrator` with RWX permission

The ruleData shall point to the registered Log MO and everything beneath it

See Example 1, `saLogSystem` is a system log stream registered with the Log Management Framework, and the following 2 rules are delivered during installation.



```

<object parentDn="ManagedElement=1, SystemFunctions=1, SecM=1, UserManagement=1, LocalAuthorizationMethod=1,
Role=SystemReadOnly">
  <hasClass name="Rule">
    <mimName>ECIM_Local_Authorization</mimName>
  </hasClass>
  <slot name="ruleId">
    <value>LogManagement_4</value>
  </slot>
  <slot name="permission">
    <value>R</value>
  </slot>
  <slot name="ruleData">
    <value>ManagedElement, SystemFunctions, LogM, Log=saLogSystem, *</value>
  </slot>
  <slot name="userLabel">
    <value/>
  </slot>
</object>
</slot>

<object parentDn="ManagedElement=1, SystemFunctions=1, SecM=1, UserManagement=1, LocalAuthorizationMethod=1,
Role=SystemAdministrator">
  <hasClass name="Rule">
    <mimName>ECIM_Local_Authorization</mimName>
  </hasClass>
  <slot name="ruleId">
    <value>LogManagement_5</value>
  </slot>
  <slot name="permission">
    <value>RWX</value>
  </slot>
  <slot name="ruleData">
    <value>ManagedElement, SystemFunctions, LogM, Log=saLogSystem, *</value>
  </slot>
  <slot name="userLabel">
    <value/>
  </slot>
</object>

```

Example 1 Security Rule Objects for saLogSystem

5.2.4 Severity Filter Must Align with Log and Trace Design Rules

The Log and Trace Design Rules have the following impact on the Log Management Framework.

1. It is not allowable to mask/disable EMERGENCY, ALERT and CRITICAL level of logs.
2. The recommended default severity filter level is NOTICE level and above.
3. INFO level of logging and higher is allowed for troubleshooting.

The Log Management Framework will enforce rules relating to allowed and disallowed severity filters – for example rule #1 above.

It is a recommendation for Log Stream Owners, when designing a Log MO for their registered log streams, to ensure that the initial value of the severity filter bitmask is NOTICE and higher.



5.3 Registering SAFLOG Log Streams

The only type of SAFLOG stream that can be registered by the LogM framework is SAFLOG Configuration Log streams. SAFLOG Runtime streams cannot be registered by the LogM framework.

5.3.1 Opening SAFLOG Configuration Log Streams

For legacy applications which are Log Stream Owners which open SAFLOG Application Runtime streams using the OpenSAF Log Service API, the method of creating the log stream must change so that a SAFLOG Configuration stream is opened instead.

Once a SAFLOG log stream has been created as a Configuration stream, an application can use the OpenSAF Log Service API to log entries in the same manner as a Runtime stream.

To create a Configuration stream, an application which is a Log Stream Owner needs to be required to design and deploy a `SaLogStreamConfig` object for each log stream they own in order for them to be registered with the LogM framework.

The application developer must deploy the objects with the application installation package.

For detailed description of the `SaLogStreamConfig` class definition, refer to Section 4.2 Log Service UML Classes in *SA Forum Application Interface Specification: Log Service (LOG)*.

An example `SaLogStreamConfig` instance file is also provided in the Core MW LogM SDK.