Working Draft
MEF W146 v0.1

LSO Allegro, LSO Interlude and LSO Legato Alarms and Threshold Crossing Alerts API - Developer Guide

This draft represents MEF work in progress and is subject to change.

December 2024

Disclaimer

The information in this publication is freely available for reproduction and use by any recipient and is believed to be accurate as of its publication date. Such information is subject to change without notice and MEF Forum (MEF) is not responsible for any errors. MEF does not assume responsibility to update or correct any information in this publication. No representation or warranty, expressed or implied, is made by MEF concerning the completeness, accuracy, or applicability of any information contained herein and no liability of any kind shall be assumed by MEF as a result of reliance upon such information.

The information contained herein is intended to be used without modification by the recipient or user of this document. MEF is not responsible or liable for any modifications to this document made by any other party.

The receipt or any use of this document or its contents does not in any way create, by implication or otherwise:

- (a) any express or implied license or right to or under any patent, copyright, trademark or trade secret rights held or claimed by any MEF member which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor
- (b) any warranty or representation that any MEF member will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor
- (c) any form of relationship between any MEF member and the recipient or user of this document.

Implementation or use of specific MEF standards, specifications or recommendations will be voluntary, and no Member shall be obliged to implement them by virtue of participation in MEF Forum. MEF is a non-profit international organization to enable the development and worldwide adoption of agile, assured and orchestrated network services. MEF does not, expressly or otherwise, endorse or promote any specific products or services.

Copyright

Table of Contents

- List of Contributing Members
- 1. Abstract
- 2. Terminology and Abbreviations
- 3. Compliance Levels
- 4. Introduction
  - 4.1. Description
  - 4.2. Conventions in the Document
  - 4.3. Relation to Other Documents
  - 4.4. Approach
  - 4.5. High-Level Flow
- 5. API Description
  - 5.1. High-level use cases
  - 5.2. API Endpoint and Operation Description
    - 5.2.1. Seller/Server side API Endpoints
    - 5.2.2. Buyer/Client side API Endpoints
  - 5.3. Integration of Alarm Specification into Alarm Management API
  - 5.4. Model structure and validation
  - 5.5. Security Considerations
- 6. API Interactions and Flows
  - 6.1. Use Case 1: Retrieve List of Alarms
  - 6.2. Use Case 2: Retrieve Alarm by Alarm Identifier
  - 6.3. Use Case 3: Subscribe to Alarms
  - 6.4. Use Case 4: Create Alarm
    - 6.4.1 Use Case 4a: Stateful TCA Alarm
    - 6.4.2 Use Case 4b: Stateless TCA Alarm
  - 6.5. Use Case 5: Unregister for Alarm Notifications
- 7. API Details
  - 7.1. API patterns
    - 7.1.1. Indicating errors
      - 7.1.1.1. Type Error
      - 7.1.1.2. Type Error400
      - 7.1.1.3. `enum` Error400Code
      - 7.1.1.4. Type Error401
      - 7.1.1.5. `enum` Error401Code
      - 7.1.1.6. Type Error403
      - 7.1.1.7. `enum` Error403Code
      - 7.1.1.8. Type Error404
      - 7.1.1.9. Type Error422
      - 7.1.1.10. `enum` Error422Code
      - 7.1.1.11. Type Error500
      - 7.1.1.12. Type Error501
  - 7.2. API Data model
    - 7.2.1 Alarm
      - 7.2.1.1 Type Alarm
      - 7.2.1.2 Type AlarmRef
      - 7.2.1.3 Type AlarmSpecificAttributes
      - 7.2.1.4 `enum` AlarmType
      - 7.2.1.5 Type Alarm_Common
      - 7.2.1.6 Type Alarm_Find
      - 7.2.1.7 Type AlarmedObjectRef
      - 7.2.1.8 Type Comment
      - 7.2.1.9 `enum` PerceivedSeverity

        - 7.2.1.10 enum PlannedOutageIndicator
        - 7.2.1.11 enum ProbableCause
        - 7.2.1.12 Type ServiceRef
        - 7.2.1.13 enum State
    - 7.2.2 TCA Alarm
        - 7.2.2.1 Type TcaStatefulClearAlarm
        - 7.2.2.2 Type TcaStatefulSetAlarm
        - 7.2.2.3 Type TcaStatelessAlarm
    - 7.2.3. Notification registration
        - 7.2.3.1. Type AlarmSubscriptionInput
        - 7.2.3.2. Type AlarmSubscription
- 8. References

List of Contributing Members

The following members of the MEF participated in the development of this document and have requested to be included in this list.

Member

Table 1. Contributing Members

1. Abstract

This standard is intended to assist the implementation of the Application Programming Interfaces (APIs) for the Alarm ManagementAPI functionality of the Service Orchestration Function at the LSO Allegro, LSO Interlude and LSO Legato Interface Reference Points (IRPs), for which requirements and use cases are defined in MEF W133.1 [MEF W133.1]. The requirements and use cases are the same for all IRPs. This standard consists of this document and complementary API definitions for Service Funtion Testing Management and Service Function Testing Notifications.

This standard normatively incorporates the following files by reference as if they were part of this document from the GitHub repository:

MEF-LSO-Allegro-SDK

- `serviceApi/alarm/alarmManagement.api.yaml`
- `serviceApi/alarm/alarmNotification.api.yaml`

MEF-LSO-Interlude-SDK

- `serviceApi/alarm/alarmManagement.api.yaml`
- `serviceApi/alarm/alarmNotification.api.yaml`

MEF-LSO-Legato-SDK

- `serviceApi/alarm/alarmManagement.api.yaml`
- `serviceApi/alarm/alarmNotification.api.yaml`

The Alarm Management API is defined using OpenAPI 3.0 [OAS-V3]

2. Terminology and Abbreviations

This section aims to clarify the terminology used throughout this document. In many cases, the authoritative definitions of terms can be found in separate documents. To ensure accuracy and consistency, the third column of this document serves to provide the appropriate references from MEF or external sources that govern these definitions.

In addition, terms defined in the standards referenced below are included in this document by reference and are not repeated in the table below:

- MEF W133.1 *Allegro, Interlude and Legato Alarm Management BR&UC* [MEF W133.1]
- MEF 55.1 *Lifecycle Service Orchestration (LSO): Reference Architecture and Framework* [MEF 55.1]

Term	Definition	Source
Alarm	A specific type of notification concerning detected faults or abnormal conditions.	MEF W133.1
API Endpoint	The endpoint of a communication channel (the complete URL of an API Resource) to which the HTTP-REST requests are addressed to operate on the *API Resource*.	rapidapi.com This document
API Resource	A REST Resource. In REST, the primary data representation is called Resource. In this document, *API Resource* is defined as an OAS *SchemaObject* with specified *API Endpoints*.	restfulapi.net This document
Notification	In general, a mechanism used to inform the recipient about certain event in the system. In context of this document notification is a synchronous communication from the observed system towards recipient	MEF W133.1
OpenAPI	The OpenAPI 3.0 Specification, formerly known as the Swagger specification is an API description format for REST APIs.	spec.openapis.org
Operation	An interaction between the Server and Client, potentially involving multiple back-and-forth transactions.	This document
REST API	Representational State Transfer. REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems.	REST API
SchemaObject	The construct that allows the definition of input and output data types. These types can represent object classes, as well as primitives and array specifications.	spec.openapis.org
Threshold Crossing Alert	Mechanism used to monitor and notify when specific thresholds or performance limits are exceeded or crossed.	MEF W133.1

Table 2. Terminology

Term	Definition	Source

Term	Definition	Source
API	Application Programming Interface. In this document, API is used synonymously with REST API.	This document
Buyer/Client	Business Applications	MEF 55.1
CUS	Customer Application Coordinator	MEF 55.1
IRP	Interface Reference Point	This document
OAS	OpenAPI Specification	openapis.org
TCA	Threshold Crossing Alert	MEF W133.1
Seller/Server	Service Orchestration Functionality	MEF 55.1

Table 3. Abbreviations

3. Compliance Levels

The key words **"MUST"**, **"MUST NOT"**, **"REQUIRED"**, **"SHALL"**, **"SHALL NOT"**, **"SHOULD"**, **"SHOULD NOT"**, **"RECOMMENDED"**, **"NOT RECOMMENDED"**, **"MAY"**, and **"OPTIONAL"** in this document are to be interpreted as described in BCP 14 (RFC 2119 [RFC2119], RFC 8174 [RFC8174]) when, and only when, they appear in all capitals, as shown here. All key words must be in bold text.

Items that are **REQUIRED** (contain the words **MUST** or **MUST NOT**) are labeled as **[Rx]** for required. Items that are **RECOMMENDED** (contain the words **SHOULD** or **SHOULD NOT**) are labeled as **[Dx]** for desirable. Items that are **OPTIONAL** (contain the words MAY or OPTIONAL) are labeled as **[Ox]** for optional.

A paragraph preceded by **[CRa]<** specifies a conditional mandatory requirement that **MUST** be followed if the condition(s) following the "<" have been met. For example, **"[CR1]<[D38]"** indicates that Conditional Mandatory Requirement 1 must be followed if Desirable Requirement 38 has been met. A paragraph preceded by **[CDb]<** specifies a Conditional Desirable Requirement that **SHOULD** be followed if the condition(s) following the "<" have been met. A paragraph preceded by **[COc]<**specifies a Conditional Optional Requirement that **MAY** be followed if the condition(s) following the "<" have been met.

4. Introduction

The Alarm Management API allows the Buyer to retrieve Alarms as well as receive notifications An alarm is a signal or notification designed to alert listening client of condition or event that requires attention or intervention. The alarm can indicate that a TCA has been crossed, which is independent of the state of the service.

This standard specification document describes the Application Programming Interface (API) for Alarm Managment functionality of the LSO Allegro Interface Reference Point (IRP), LSO Interlude Interface Reference Point (IRP) and LSO Sonata IRP as defined in the *MEF 55.1 Lifecycle Service Orchestration (LSO): Reference Architecture and Framework* [MEF55.1]. The LSO Reference Architecture is shown in Figure 1 with the three IRPs highlighted.

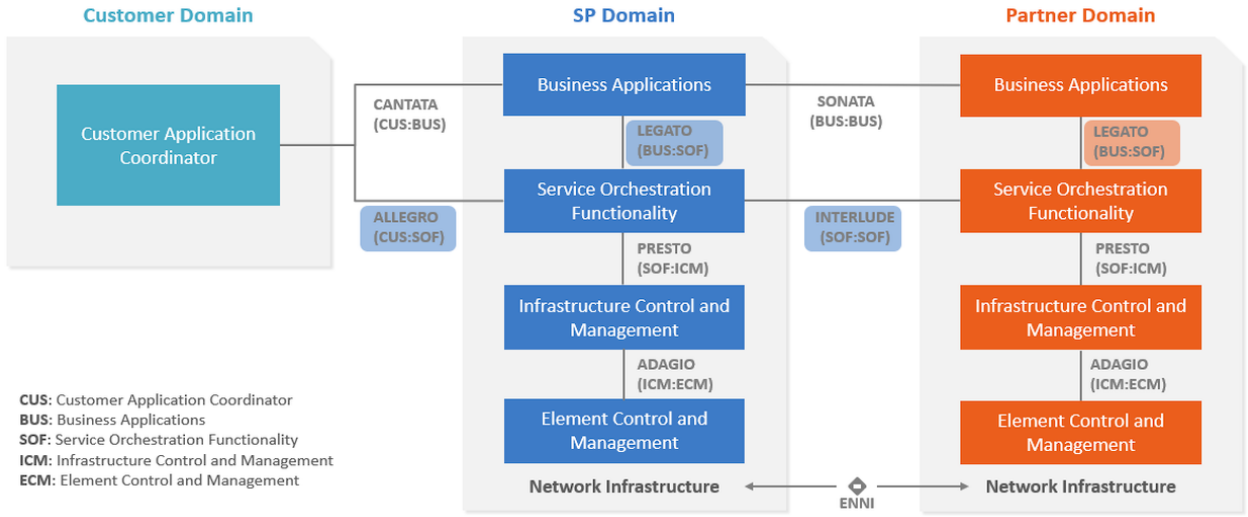

Figure 1. The LSO Reference Architecture

Note: The use cases and business requirements in this document assume a two-actor relationship based on the set of actors in the LSO architecture. The names of the relationships are specific to the Interface Reference Point. For both Allegro and Interlude there is a Buyer and Seller. For Allegro the Buyer is the Customer and the Seller is the Service Provider. In Interlude the Buyer is the Service Provider and the Seller is the Partner. In the case of the Legato IRP, given this is within a single Service Provider or Partner, the relationship is between Client and Server, where the Business Application (BA) is the Client, and the Service Orchestration Functionality is the Server. Considering this duality, actors in the document are referred to as Buyer/Client and Seller/Server.

4.1. Description

The scope of this API and Developer Guide covers

- Alarm Managment
  - Includes retrieval of Alarms
- Alarm Notification
  - Includes Event Subscription/Hub and Listener notification functions

The business requirements and use cases for Alarm Management are defined in MEF W133.1 Allegro, Interlude and Legato Fault Management and Performance Monitoring BR&UC MEFW133.1.

This document supports interactions over the Legato interface within a single operator as well as interaction with Partner Domain and Customer Domain through Interlude and Allegro interfaces

respectively.

Business Applications (Buyer/Client), Customer Application Coordinator (CUS) and Service Orchestration Functionality systems use the information contained within this document.

This standard is intended to support the design of API implementations that enable interoperable Seller/Server operations (in the scope of this standard) across the Allegro IRP, Interlude IRP, and Legato IRP.

4.2. Conventions in the Document

- Code samples are formatted using code blocks. When notation `<< some text >>` is used in the payload sample it indicates that a comment is provided instead of an example value, and it might not comply with the OpenAPI definition.
- Model definitions are formatted as in-line code (e.g. `PerformanceJob`).
- In UML diagrams the default cardinality of associations is `0..1`. Other cardinality markers are compliant with the UML standard.
- In the API details tables and UML diagrams required attributes are marked with a `*` next to their names.
- In UML sequence diagrams `{{variable}}` notation is used to indicate a variable to be substituted with a correct value.

4.3. Relation to Other Documents

This API implements the Alarm Managment related requirements and use cases that are defined in MEF W133.1 [MEFW133.1]. The API definition builds on TMF Open API (v5.0.0) for Alarm Management API TMF 642.

4.4. Approach

As presented in Figure 2. the Allegro, Interlude, and Legato API frameworks consist of three structural components:

- Generic API framework
- Service-independent information (Function-specific information and Function-specific operations)
- Service-specific information (MEF service specification data model)

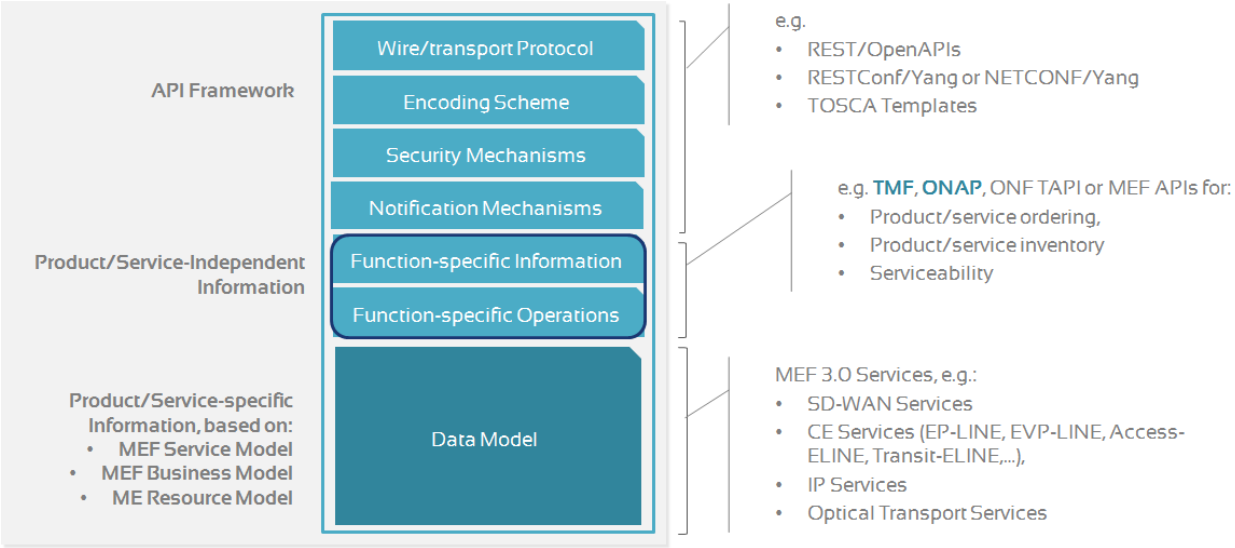

Figure 2. Allegro, Interlude and Legato API Structure

The essential concept behind the framework is to decouple the common structure, information, and operations from the specific alarm information content. Firstly, the Generic API Framework defines a set of design rules and patterns that are applied across all Allegro, Interlude, and Legato APIs. Secondly, the alarm-independent information of the framework focuses on a model of a particular Allegro, Interlude, or Legato functionality and is agnostic to any of the alarm specifications. For example, this standard is describing the Alarm Model and operations that allow managment of alarms that are aligned with either MEF or custom alarm specifications

This Developer Guide is not defining MEF alarm specifications but can be used in combination with any alarm specifications defined by or compliant with MEF

Figure 3 presents the relationship between the Alarm Managemnt API entities and the alarm specification model. The `alarmSpecificAttributes` serves as an extension point for configuring alarm-specific parameters.

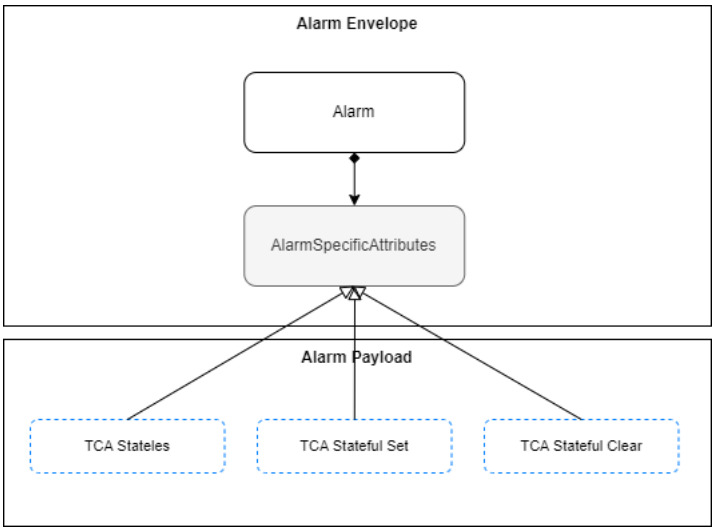

Figure 3. Alarm specification for Allegro, Interlude, Legato

4.5. High-Level Flow

The Alarm Management API allows the Buyer to retrieve Alarms as well as receive notifications when new Alarm is created. This allows timely detection and resolution of faults for a Alarms provided by the Seller.

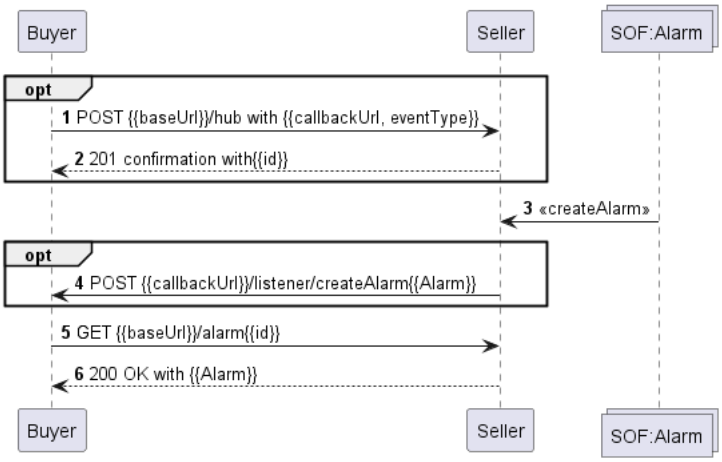

Figure 4. High-Level Flow

The following steps describe the high-level flow:

- The Buyer/Client registers listeners for notifications related to alarms via the Hub.
  Note1: Alarm Notifications are optional and do not impact end-to-end flow
- When fault or abnormal codition is identified the SOF creates an `Alarm` in the Server/Seller system
- (optional) The Seller/Server sends the `Alarm` notification to a buyer.
- The Buyer/Client system retrieves `Alarm` through *Alarm API*

5. API Description

This section presents the API structure and design patterns. It starts with the high-level use cases diagram. Then it describes the REST endpoints with use case mapping. Next, it explains the design pattern that is used to combine alarm-agnostic and alarm-specific parts of API payloads. Finally, payload validation and API security aspects are discussed.

5.1. High-level use cases

Figure 5 presents a high-level use case diagram. It aims to help understand the endpoint mapping. Use cases are described extensively in chapter 6.

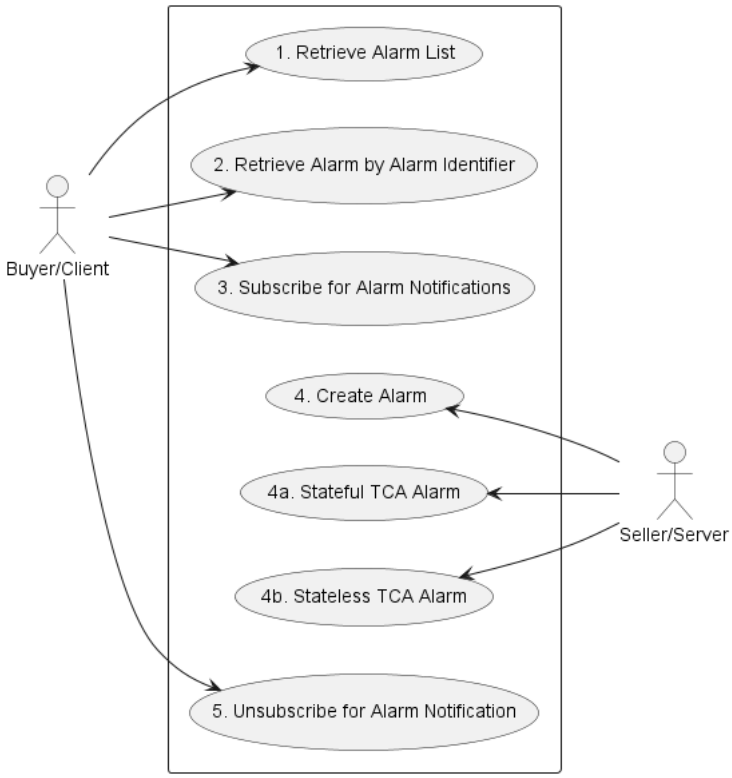

Figure 5. Use cases

5.2. API Endpoint and Operation Description

5.2.1. Seller/Server side API Endpoints

Base URL for Allegro:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/allegro/alarmManagement/v1/`

Base URL for Interlude:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/interlude/alarmManagement/v1/`

Base URL for Legato:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/legato/alarmManagement/v1/`

The following API endpoints are implemented by the Seller/Server and allow the Buyer/Client to retrieve `Alarm` instances. The endpoints and corresponding data model are defined in `serviceApi/alarm/alarmManagement.api.yaml`.

API Endpoint	Description	MEF W133.1 Use Case Mapping
`GET /alarm`	The Buyer/Client requests a list of Alarms based on a set of filter criteria.	UC 49: Retrieve Alarm List
`GET /alarm/{{id}}`	The Buyer/Client requests detailed information about a single Alarm	UC 50: Retrieve Alarm by Identifier

Table 4. Seller/Server mandatory API endpoints

[R1] Seller/Server **MUST** support all API endpoints listed in Table 4.

API endpoints listed in Table 5 are optional and may be exposed by the Seller/Server.

API Endpoint	Description	MEF W133.1 Use Case Mapping
`POST /hub`	The Buyer/Client or Administrator requests to subscribe to the Alarm Notifications.	UC 51: Subscribe to Alarms
`GET /hub/{{id}}`	The Buyer/Client or Administrator retrieves a specific `EventSubscription` from the Seller/Server, that matches the *`id`* value provided as *`path`* parameter.	
`DELETE /hub/{{id}}`	The Buyer/Client or Administrator requests to unsubscribe from the Alarm Notifications.	UC 52: Un-Subscribe from Alarms

Table 5. Seller/Server optional API endpoints

[O1] The implementation **MAY** support API endpoints listed in Table 5.

5.2.2. Buyer/Client side API Endpoints

Base URL for Allegro:

```
https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/allegro/alarmNotification/v1/
```

Base URL for Interlude:

```
https://{{serverBase}}:{{port}}
{{?/sof_prefix}}/mefApi/interlude/alarmNotification/v1/
```

Base URL for Legato:

```
https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/legato/alarmNotification/v1/
```

The following API Endpoints are used by Seller/Server to post notifications to registered Buyer/Client listeners. The endpoints and corresponding data model are defined in `serviceApi/alarm/alarmNotification.api.yal`

API Endpoint	Description	MEF W133.1 Use Case Mapping

API Endpoint	Description	MEF W133.1 Use Case Mapping
`POST /listener/createAlarm`	A request initiated by the Seller/Server to notify Buyer/Client on `Alarm` instance creation.	UC 48: Create Alarm Notification, UC 53: Stateful TCA Alarm, UC 54: Stateless TCA Alarm

Table 6. Buyer/Client API endpoints

[O2] The Buyer/Client **MAY** support API endpoints listed in Table 6.

[O3] The Buyer/Client **MAY** register to receive Alarm Notifications.

[R2] The Seller/Server **MUST** support sending notifications to API endpoints listed in Table 6 to the registered Buyer/Client.

5.3. Integration of Alarm Specification into Alarm Management API

Alarm Management API data model discussed in this document is a generic envelope that allows for the management of relevant Alarm objects. The API data model itself does not provide explicit definitions or prescribing the structure for different types of alarm. However, it offers flexible extensibility to accommodate the configuration of alarm-specific objectives. This allows for customization and adaptation to various requirements. Alarm Management API schema is defined using JsonSchema (draft 7) format JSON Schema draft 7 and can be integrated into the `Alarm` using the TMF extension pattern.

The extension hosting type in the API data model is:

- `AlarmSpecificAttributes` - this type is extended with Alarm Specific attributes that define how a Test is performed for a given Test Specification.

The `@type` attribute of those extension hosting types must be set to a value that uniquely identifies the service testing configuration. A unique identifier for MEF standard service schemas is in URN format and is assigned by MEF. This identifier is provided as root schema `$id`. Use of non-MEF standard service testing configuration is allowed. In such a case the schema identifier must be agreed upon between the Buyer/Client and the Seller/Server.

The example below shows a header of a schema, which describes the TCA Stateless Alarm Configuration configuration, where `urn:mef:lso:spec:legato:tca-stateless-alarm:v0.0.1:all` is the above-mentioned URN:

```
"$schema": "http://json-schema.org/draft-07/schema#"
"$id": "urn:mef:lso:spec:legato:tca-stateless-alarm:v0.0.1:all"
title: TCA Stateless Alarm Configuration
```

Alarm specific configuration payload is introduced in Alarm Management API entities through a `alarmSpecificAttributes` attribute of type `AlarmSpecificAttributes` which is used as an extension point for configuration attributes.

Implementations might choose to integrate selected Alarm Management API specifications to data model during development e.g. TCA Alarm. In such a case an integrated data model is built, and alarm specifications are in an inheritance relationship with either `AlarmSpecificAttributes` as described in the OAS specification. This pattern is called **Static Binding**. The snippets below present an example of a static binding of the envelope API with exemplary MEF TCA Alarm specifications.

```
AlarmSpecificAttributes:
  type: object
  description: AlarmSpecificAttributes is used as an extension for Alarm specific
    payload.  The @type attribute is used as a discriminator.
  discriminator:
    mapping:
      urn:mef:lso:spec:legato:tca-stateful-set-alarm:v0.0.1:all: '#/components/schemas/TcaStatefulSetAlarm'
      urn:mef:lso:spec:legato:tca-stateless-alarm:v0.0.1:all: '#/components/schemas/TcaStatelessAlarm'
      urn:mef:lso:spec:legato:tca-stateful-clear-alarm:v0.0.1:all: '#/components/schemas/TcaStatefulClearAlarm'
    propertyName: '@type'
  properties:
    '@type':
      type: string
      description: The named type must be a subclass of AlarmSpecificAttributes.
      enum:
      - urn:mef:lso:spec:legato:tca-stateful-set-alarm:v0.0.1:all
      - urn:mef:lso:spec:legato:tca-stateless-alarm:v0.0.1:all
      - urn:mef:lso:spec:legato:tca-stateful-clear-alarm:v0.0.1:all
  required:
  - '@type'
```

```
TcaStatelessAlarm:
  allOf:
  - $ref: '#/components/schemas/AlarmSpecificAttributes'
  - type: object
    description: Threshold Crossing Alert Alarm Schema.
```

Alternatively, implementations might choose not to build an integrated model and choose a different mechanism allowing runtime validation of alarm-specific fragments of the payload. The system can validate a given monitoring configuration against a new schema without redeployment. This pattern is called **Dynamic Binding.**

Regardless of the chosen implementation pattern, the HTTP payload is the same. Both implementation approaches must conform to the requirements specified below.

[R3] `AlarmSpecificAttributes` type is extension points that **MUST** be used to integrate alarm specific properties into a request/response payload.

[R4] The `@type` property of `AlarmSpecificAttributes` **MUST** be used to specify the type of the extending entity.

[R5] Attributes specified in the payload must conform to the alarm definition specified in the `@type` property.

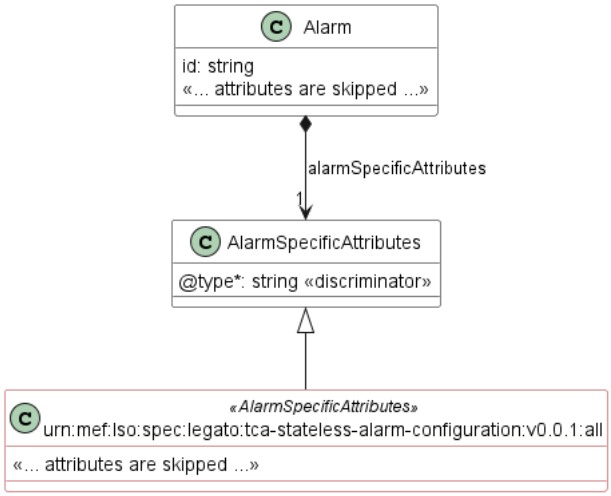

Figure 6. The Extension Pattern with Sample Alarm-Specific Extension

Figure 6 presents two MEF Alarm Management API schemas that represent configuration and result classes for IP services. When these schemas are used, the `@type` of `AlarmSpecificAttributes` takes `"urn:mef:lso:spec:legato:tca-stateless-alarm:v0.0.1:all"` value to indicate which test specification should be used to interpret a set of alarm-specific attributes included in the payload.

5.4. Model structure and validation

The structure of the payloads exchanged via Allegro, Interlude, and Legato Alarm Management API endpoints is defined using:

- OpenAPI version 3.0 for the service-agnostic part of the payload
- JsonSchema (draft 7) for the alarm-specific part of the payload

[R6] Implementations **MUST** use payloads that conform to these definitions.

5.5. Security Considerations

Although the Legato IRP is internal to a Service Provider/Operator business boundary, it is expected that some minimal security mechanisms are in place for any communication over this IRP. There must also be authorization mechanisms in place to control what a particular Buyer/Client or Seller/Server is allowed to do and what information may be obtained. For Allegro and Interlude IRPs, security should follow rules for external communication. The definition of the exact security mechanism and configuration is outside the scope of this document. The LSO Security mechanisms are defined by MEF 128.1 *LSO API Security Profiles* [MEF128.1].

6. API Interactions and Flows

This section provides a detailed insight into the API functionality, use cases, and flows. It starts with Table 7 presenting a list and short description of all business use cases then present the variants of end-to-end interaction flows, and in the following subchapters describe the API usage flow and examples for each of the use cases.

Use Case #	Use Case Name	Use Case Description
1	Retrieve Alarm List	The Buyer/Client requests a list of Alarms based on a set of filter criteria. The Seller/Server returns a summarized list of Alarms.
2	Retrieve Alarm by Alarm Identifier	The Buyer/Client requests detailed information about a single Alarm based on the Alarm Identifier.
3	Subscribe to Alarm Notifications	The Buyer/Client requests to subscribe to Alarm Notifictions.
4	Create an Alarm	A Seller/Server sends a Create Alarm to the Buyer/Client based on an event that has occurred.
4a	Stateful TCA Alarm	A Stateful TCA Alarm is initiated by the Seller/Server to a subscribed Client.
4b	Stateless TCA Alarm	A Stateless TCA lifecycle alarm is initiated by the Seller/Server to a subscribed Client.
5	Usubscribe for Alarm Notifications	The Buyer/Client requests to unsubscribe to Alarm and/or Test Job Notifications.

Table 7. Use cases description

6.1. Use Case 1: Retrieve List of Alarms

The Buyer/Client can retrieve a list of `Alarm_Find` by using a `GET /alarm` operation with desired filtering criteria.

[R7] The Buyer/Client Retrieve List of Alarms request **MUST** contain none or more of the following attributes as filter criteria: [MEFW133.1 R138]

- `id`
- `alarmDetails`
- `alarmChangedTime.gt`
- `alarmChangedTime.lt`
- `alarmClearedTime.gt`
- `alarmClearedTime.lt`
- `alarmReportingTime.gt`
- `alarmReportingTime.lt`
- `alarmType`
- `alarmedObjectType`
- `perceivedSeverity`
- `plannedOutageIndicator`
- `reportingSystemId`

- `serviceAffecting`
- `state`
- `affectedServiceId`
- `correlatedAlarmId`

```
https://serverRoot/mefApi/legato/alarmManagement/v1/alarm?alarmChangedTime.gt="2024-08-
12T23:20:50.52Z"&limit=10&offset=0
```

The example above shows a Buyer/Client's request to get all Alarms objects created after `2024-08-12T23:20:50.52Z`. Additionally, the Buyer/Client asks only for a first (`offset=0`) pack of 10 results (`limit=10`) to be returned. The correct response (HTTP code `200`) in the response body contains a list of `Alarm_Find` objects matching the criteria. To get all the details, the Buyer/Client has to query a specific `Alarm` by its `id`. Details related to pagination are described in section 7.1.2

If the quantity of the records requested to be returned exceeds a Seller/Server policy, the Seller/Server must choose to respond with either:

- An empty list and message that indicates the result set is too large or
- A response that indicates the result is too large and includes a subset of the matching Alarms.

[R8] The Seller/Server MUST support the Retrieve Alarm List Use Case. [MEFW133.1 R139]

[R9] The Buyer/Client MUST support the Retrieve Alarm List Use Case. [MEFW133.1 R140]

[R10] The Seller/Server's response to the Buyer's/Client's Retrieve List of Alarms **MUST** include the following attributes [MEFW133.1 R141]

- `id`
- `alarmDetails`
- `alarmReportingTime`
- `alarmType`
- `perceivedSeverity`
- `state`

[O4] The Seller response MAY contain any of the remaining Alarm attributes [MEFW133.1 O25]

[R11] In case no items matching the criteria are found, the Seller/Server **MUST** return a valid response with an empty list.

[R12] If the request is unsuccessful, the Seller/Server MUST return an error with explanation to the Buyer/Client.

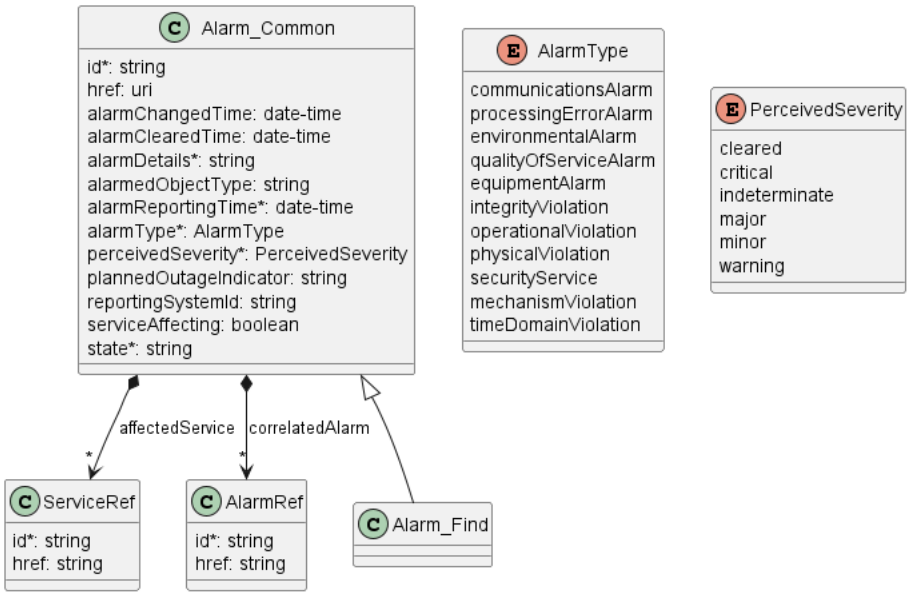

Figure 7. Use Case 1: Retrieve Alarm List - Model

6.2. Use Case 2: Retrieve Alarm by Alarm Identifier

The Buyer/Client can get detailed information about the Alarm from the Seller/Server by using a `GET /Alarm/{{id}}` operation. The payload returned in the response is a full representation of the Alarm.

[R13] The Seller/Server MUST support the Retrieve Alarm by Identifier Use Case [MEFW133.1 R142]

[R14] The Buyer/Client MUST support the Retrieve Alarm by Identifier Use Case [MEFW133.1 R143]

[R15] The Buyer's Retrieve Alarm by Alarm Identifier **MUST** include the Alarm Identifier. [MEFW133.1 R144]

[R16] The Buyer's Retrieve Alarm by Alarm Identifier **MUST NOT** include other attributes. [MEFW133.1 R145]

[R17] If the request is successful, the Seller's response to a Retrieve Alarm by Alarm Identifier request MUST include all attributes

[R18] If the request is unsuccessful, the Seller/Server MUST return an error with explanation to the Buyer/Client.

[R19] In case `id` does not allow finding a `Alarm` in Seller/Server's system, an error response `Error404` **MUST** be returned.

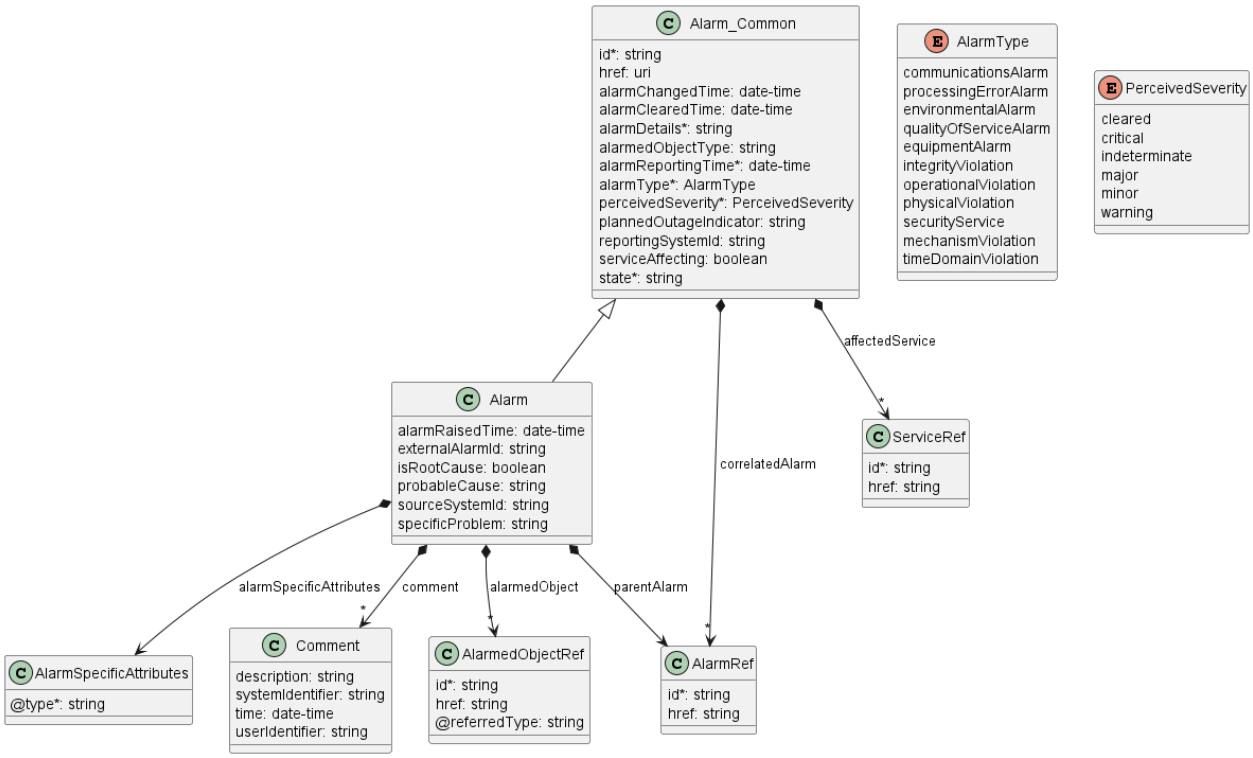

Figure 8. Use Case 2: Retrieve Alarm by Alarm Identifier - Model

6.3. Use Case 3: Subscribe to Alarms

The Buyer/Client can recive Alarms by subscribing to notifications. An exemplary use case for exchanging alarms is presented in Figure 9.

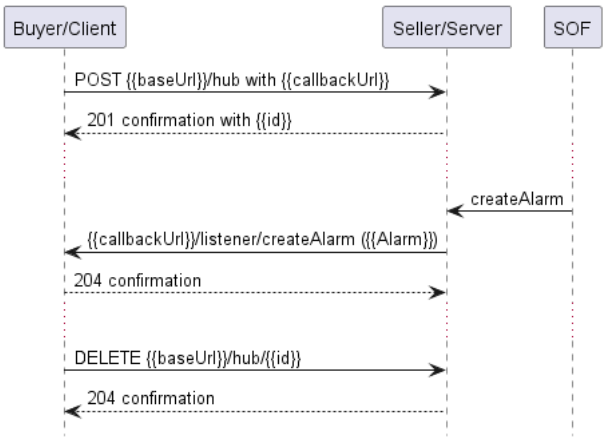

Figure 9. Alarm Notification Example

The Seller/Server communicates with the Buyer/Client with Alarm Notifications provided that:

- Buyer/Client supports a notification mechanism
- Buyer/Client has registered to receive Alarms from the Seller/Server

To register for alarms the Buyer/Client uses the `registerListener` operation from the API: `POST /hub`. The request contains 2 attributes:

- `callback` - mandatory, to provide the callback address the alarms will be notified to,
- `query`- optional, defines which type of events to register

Figure 10 shows all entities involved in the Notification use cases.

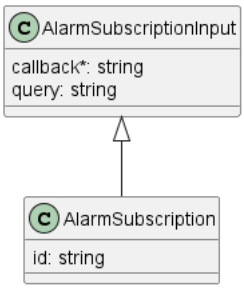

Figure 10. Alarm Management API Notification Data Model

By using a request in the following snippet, the Buyer/Client subscribes for alarm notifications

```
{
  "callback": "https://bus.com/listenerEndpoint",
  "query": "eventType = createAlarm"
}
```

[O5] The Seller/Server **MAY** support subscription to Alarms Use Case.

[O6] The Seller/Server **MAY** support unsubscribing from Alarms Use Case.

[R20] The Buyer/Client's Subscribe to Alarms request **MUST** include: [MEFW133.1 R146]

- Callback address

The Seller/Server responds to the subscription request by adding the `id` of the subscription to the message that must be further used for unsubscribing.

```
{
  "id": "00000000-0000-0000-0000-000000000678",
  "query": "eventType = createAlarm",
  "callback": "https://bus.com/listenerEndpoint"
}
```

Example of a final address that the Notifications will be sent to:

- `https://bus.com/listenerEndpoint/mefApi/legato/alarmNotification/v1/listener/createAlarm`

[R21] If successful, the Seller/Server response **MUST** indicate success and include the Register Notification Identifier and echo back all Buyer/Client provided attributes.

[R22] If successful, the Seller/Server **MUST** begin sending the alarms to the Buyer/Client.

[R23] The Seller/Server **MUST NOT** send alarms if the Buyer/Client has not registered for them.

[R24] If unsuccessful, the Seller/Server **MUST NOT** return a Register Notification Identifier.

[R25] If the Seller/Server experiences any errors, they **MUST** return an error indication to the Buyer/Client.

6.4. Use Case 4: Create Alarm

Alarm notifications are used to asynchronously inform the Buyer/Client about the detected faults or abnormal conditions.

The Figure below shows all entities involved in the Create Alarm use cases.

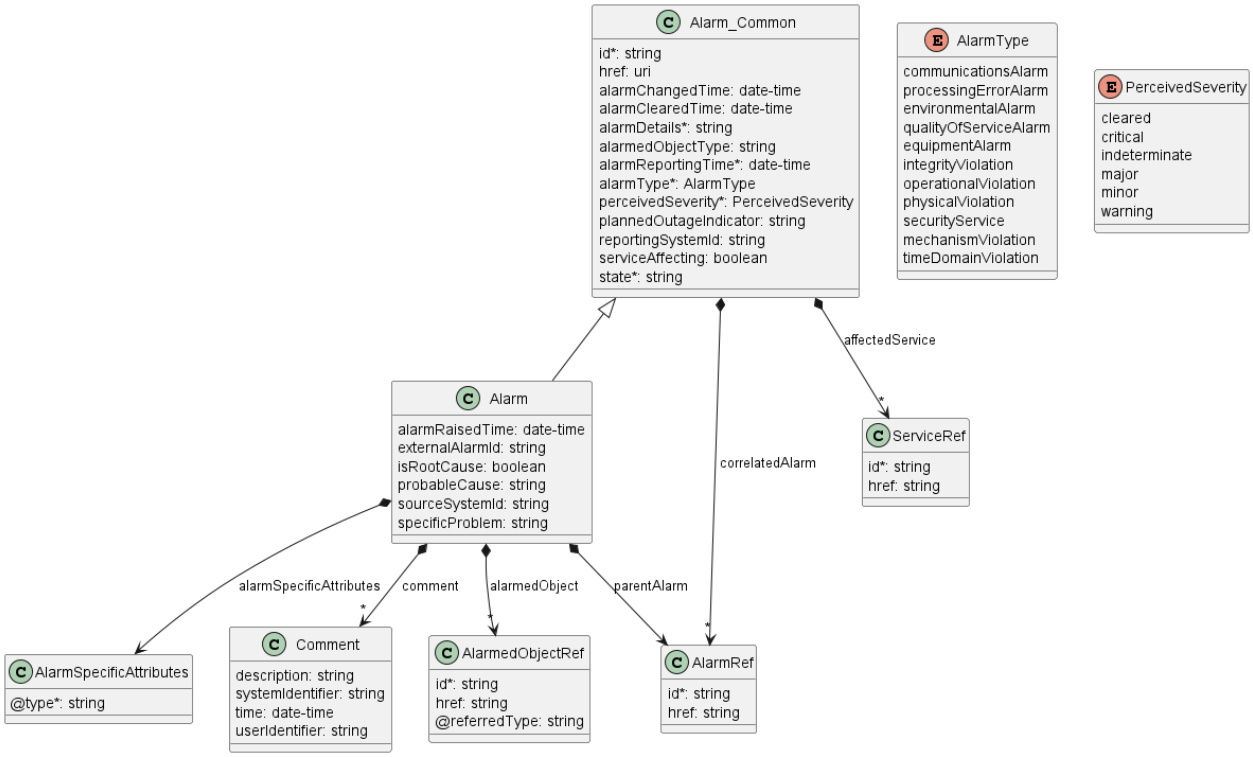

Figure 11. Use Case 4: Alarm - Model

The following snippets present an example of an `Alarm` object sent from the Seller/Server to Buyer/Client that has subscribed for Alarms.


```
{
  "id": "alarm_o1",
  "affectedService": [
    {
      "id": "service_01",
    }
  ],
  "alarmChangedTime": "2024-12-24T11:50:26.457Z",
  "alarmClearedTime": "2024-12-24T11:50:26.457Z",
  "alarmDetails": "TCA crossed",
  "alarmedObjectType": "service",
  "alarmReportingTime": "2024-12-24T11:50:26.457Z",
  "correlatedAlarm": [
    {
      "id": "string",
      "href": "string"
    }
  ],
  "alarmType": "communicationsAlarm",
  "perceivedSeverity": "cleared",
  "plannedOutageIndicator": "inPlannedMaintenance",
  "reportingSystemId": "bcc-01",
  "serviceAffecting": true,
  "state": "acknowledged",
  "alarmRaisedTime": "2024-12-24T11:50:26.457Z",
  "alarmedObject": [
    {
      "id": "port-01",
      "@referredType": "resource"
    }
  ],
  "comment": [
    {
      "description": "commnent",
      "systemIdentifier": "123",
      "time": "2024-12-24T11:50:26.457Z",
      "userIdentifier": "user-01"
    }
```

```
    ],
    "externalAlarmId": "1245",
    "isRootCause": true,
    "probableCause": "adapterError",
  }
```

[R26] The Seller/Server **MUST** include the following Alarm attributes: [MEFW133.1 R137]

- `id`
- `alarmDetails`
- `alarmReportingTime`
- `alarmType`
- `state`
- `perceivedSeverity`

[R27] The Seller/Server **MUST** send Alarms to the Buyer/Client that has registered for them.

[R28] The Seller/Server **MUST NOT** send Alarms to Buyer/Client that has not registered for them.

6.4.1 Use Case 4a: Stateful TCA Alarm

The Figure 12 shows all entities involved in Stateful TCA Alarm use case

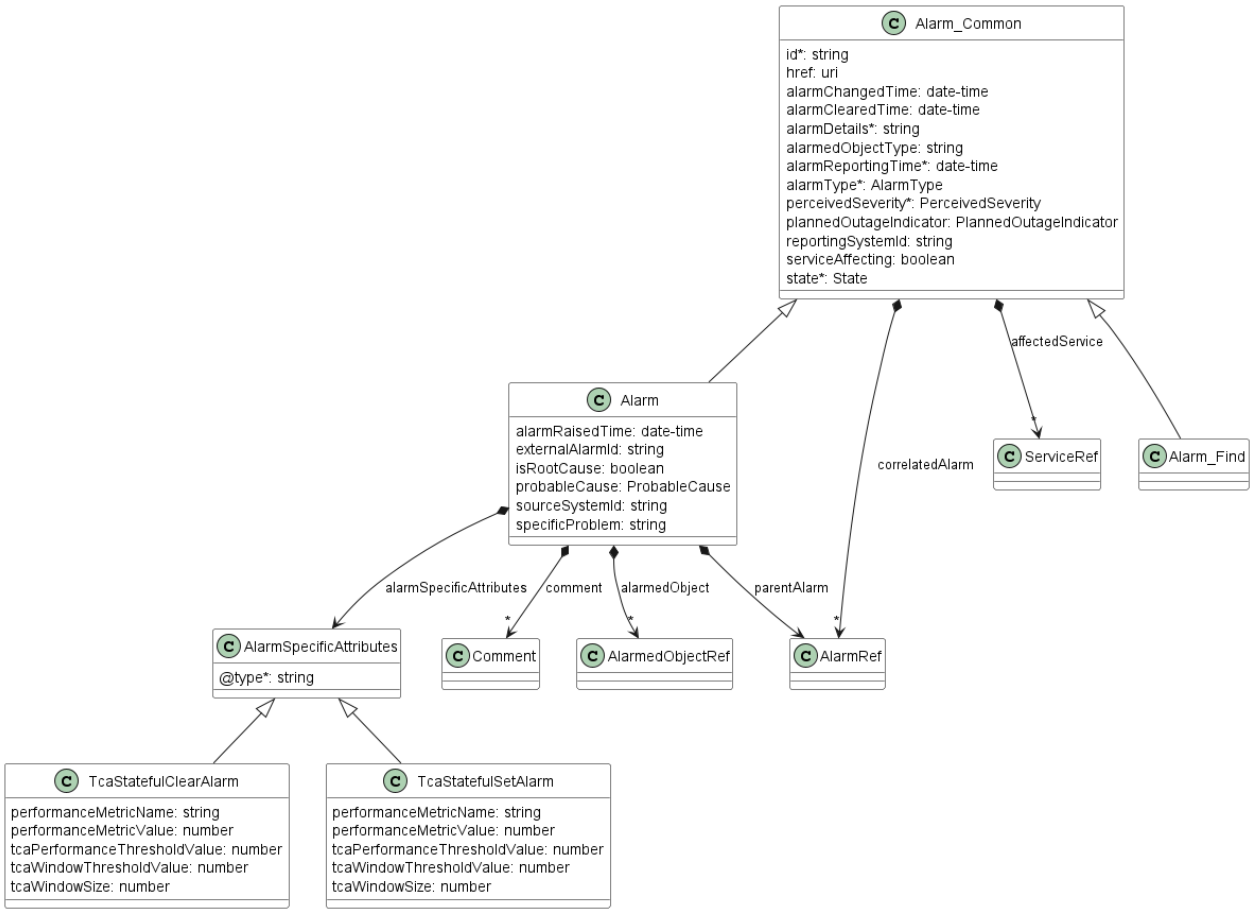

*Figure 12. Use Case 4a: Stateful TCA Alarm - Model**

[R29] When sending an alarm for a TCA Reporting Type of Stateful, the Seller/Server alarm MUST include the attributes: [MEFW133.1 R147]

- `performanceMetricName`
- `performanceMetricValue`

- `tcaPerformanceThresholdValue`
- `tcaWindowThresholdValue`
- `tcaWindowSize`

[R30] When sending an alarm for a TCA Reporting Type of Stateful, the TCA Type MUST be STATEFUL - SET when the alarm is for a TCA - SET event [MEFW133.1 R148]

[R31] When sending an alarm for a TCA Reporting Type of Stateful, the TCA Type MUST be STATEFUL - CLEAR when the alarm is for a TCA - CLEAR event. [MEFW133.1 R149]

6.4.2 Use Case 4b: Stateless TCA Alarm

The Figure 13 shows all entities involved in Stateless TCA Alarm use case

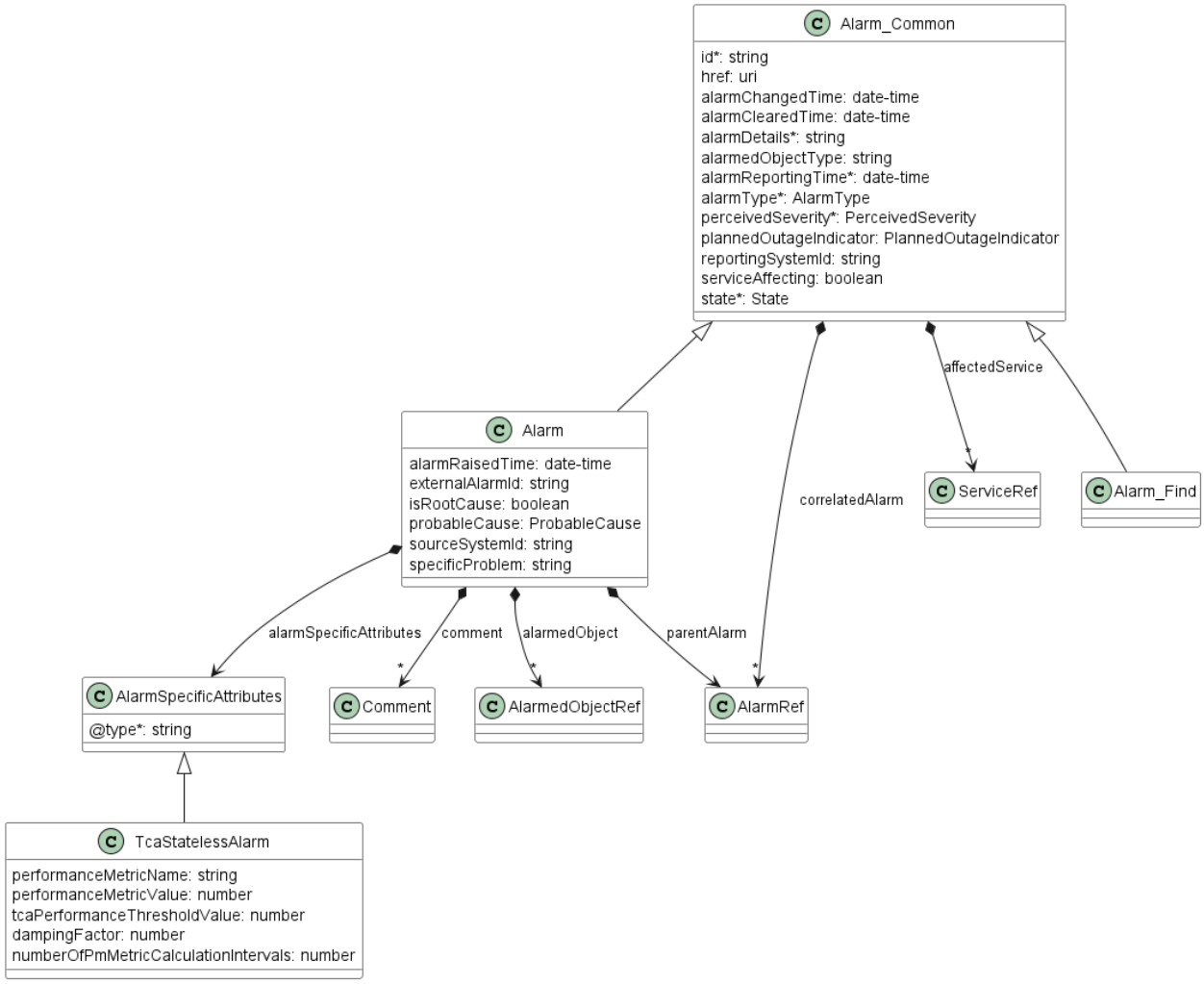

*Figure 13. Use Case 4b: Stateless TCA Alarm - Model**

[R32] When sending an alarm for a TCA Reporting Type of Stateless, the Seller/Server alarm MUST include the attributes: [MEFW133.1 R150]

- `performanceMetricName`
- `performanceMetricValue`
- `tcaPerformanceThresholdValue`
- `numberOfPmMetricCalculationIntervals`

[R33] If the Damping Factor is included in the TCA Profile, the TCA Alarm MUST include the `dampingFactor` attribute. [MEFW133.1 R151]

6.5. Use Case 5: Unregister for Alarm Notifications

To stop receiving alarms, the Buyer/Client has to use the `unregisterListener` operation from the `DELETE /hub/{id}` endpoint. The `id` is the identifier received from the Seller/Server during the listener registration.

[R34] If successful, the Seller/Server response **MUST** indicate success

[R35] If successful, the Seller/Server **MUST** stop sending the appropriate alarms to the Buyer/Client.

[R36] If unsuccessful, the Seller/Server **MUST NOT** stop sending the appropriate alarms to the Buyer/Client.

[R37] If the Seller/Server experiences any errors, they **MUST** return an error indication to the Buyer/Client.

7. API Details

7.1. API patterns

7.1.1. Indicating errors

Erroneous situations are indicated by appropriate HTTP responses. An error response is indicated by HTTP status 4xx (for client errors) or 5xx (for server errors) and appropriate response payload. The Address Validation API uses the error responses depicted and described below.

Implementations can use http error codes not specified in this standard in compliance with rules defined in RFC 7231 [RFC7231]. In such case the error message body structure might be aligned with the `Error`.

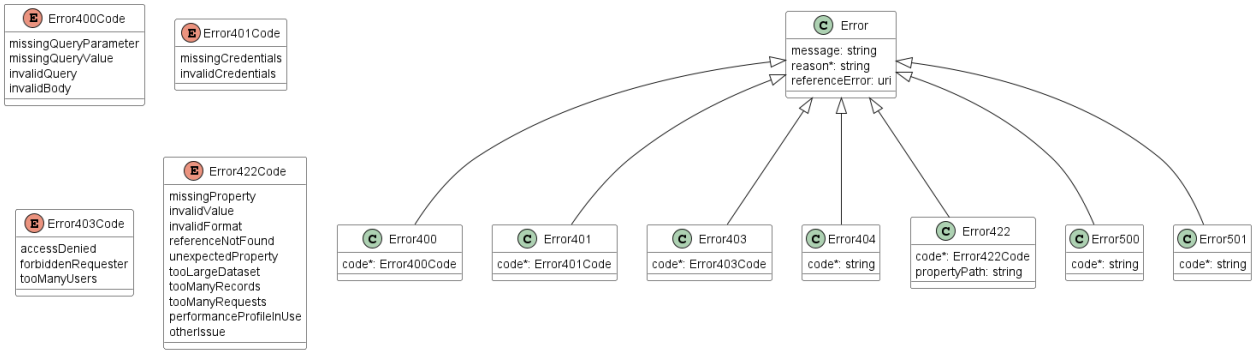

Figure 14. Data model types to represent an erroneous response

7.1.1.1. Type Error

Description: Standard Class used to describe API response error Not intended to be used directly. The `code` in the HTTP header is used as a discriminator for the type of error returned in runtime.

Name	Type	Description
message	string	Text that provides mode details and corrective actions related to the error. This can be shown to a client user.
reason*	string *maxLength = 255*	Text that explains the reason for the error. This can be shown to a client user.
referenceError	uri *format = uri*	URL pointing to documentation describing the error.

7.1.1.2. Type Error400

Description: 'Bad Request. (https://tools.ietf.org/html/rfc7231#section-6.5.1)'

Inherits from:

- Error

Name	Type	Description
code*	Error400Code	

7.1.1.3. enum Error400Code

Description: One of the following error codes:

- missingQueryParameter: The URI is missing a required query-string parameter
- missingQueryValue: The URI is missing a required query-string parameter value
- invalidQuery: The query section of the URI is invalid
- invalidBody: The request has an invalid body.

7.1.1.4. Type Error401

Description: 'Unauthorized. (https://tools.ietf.org/html/rfc7235#section-3.1)'

Inherits from:

- Error

Name	Type	Description
code*	Error401Code	

7.1.1.5. enum Error401Code

Description: One of the following error codes:

- missingCredentials: No credentials provided
- invalidCredentials: Provided credentials are invalid or expired.

7.1.1.6. Type Error403

Description: Forbidden. This code indicates that the server understood the request but refuses to authorize it. (https://tools.ietf.org/html/rfc7231#section-6.5.3)

Inherits from:

- Error

Name	Type	Description
code*	Error403Code	

7.1.1.7. enum Error403Code

Description: This code indicates that the server understood the request but refuses to authorize it because of one of the following error codes:

- accessDenied: Access denied
- forbiddenRequester: Forbidden requester
- tooManyUsers: Too many users.

7.1.1.8. Type Error404

Description: Resource for the requested path not found. (https://tools.ietf.org/html/rfc7231#section-6.5.4)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - notFound: A current representation for the target resource not found.

7.1.1.9. Type Error422

The response for HTTP status `422` is a list of elements that are structured using the `Error422` data type. Each list item describes a business validation problem. This type introduces the `propertyPath` attribute which points to the erroneous property of the request, so that the Buyer may fix it easier. It is highly recommended that this property should be used, yet remains optional because it might be hard to implement.

Description: Unprocessable entity due to a business validation problem. (https://tools.ietf.org/html/rfc4918#section-11.2)

Inherits from:

- Error

Name	Type	Description
code*	Error422Code	
propertyPath	string	A pointer to a particular property of the payload that caused the validation issue. It is highly recommended that this property should be used. Defined using JavaScript Object Notation (JSON) Pointer (https://tools.ietf.org/html/rfc6901).

7.1.1.10. `enum` Error422Code

Description: One of the following error codes:

- missingProperty: The property that was expected is not present in the payload
- invalidValue: The property has an incorrect value
- invalidFormat: The property value does not comply with the expected value format
- referenceNotFound: The object referenced by the property cannot be identified in the target system
- unexpectedProperty: Additional, not expected property has been provided
- tooLargeDataset: Requested entity will produce too many data
- tooManyRecords: The number of records to be provided in the response exceeds the threshold
- tooManyRequests: The number of simultaneous requests from one API client exceeds the threshold
- otherIssue: Other problem was identified (detailed information provided in a reason).

7.1.1.11. Type Error500

Description: Internal Server Error. (https://tools.ietf.org/html/rfc7231#section-6.6.1)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - internalError: Internal server error - the server encountered an unexpected condition that prevented it from fulfilling the request.

7.1.1.12. Type Error501

Description: Not Implemented. Used in case Seller is not supporting an optional operation (https://tools.ietf.org/html/rfc7231#section-6.6.2)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - notImplemented: Method not supported by the server.

7.2. API Data model

Figure 15 presents the whole Alarm data model. The data types, requirements related to them, and mapping to MEF W133.1 specification are discussed later in this section.

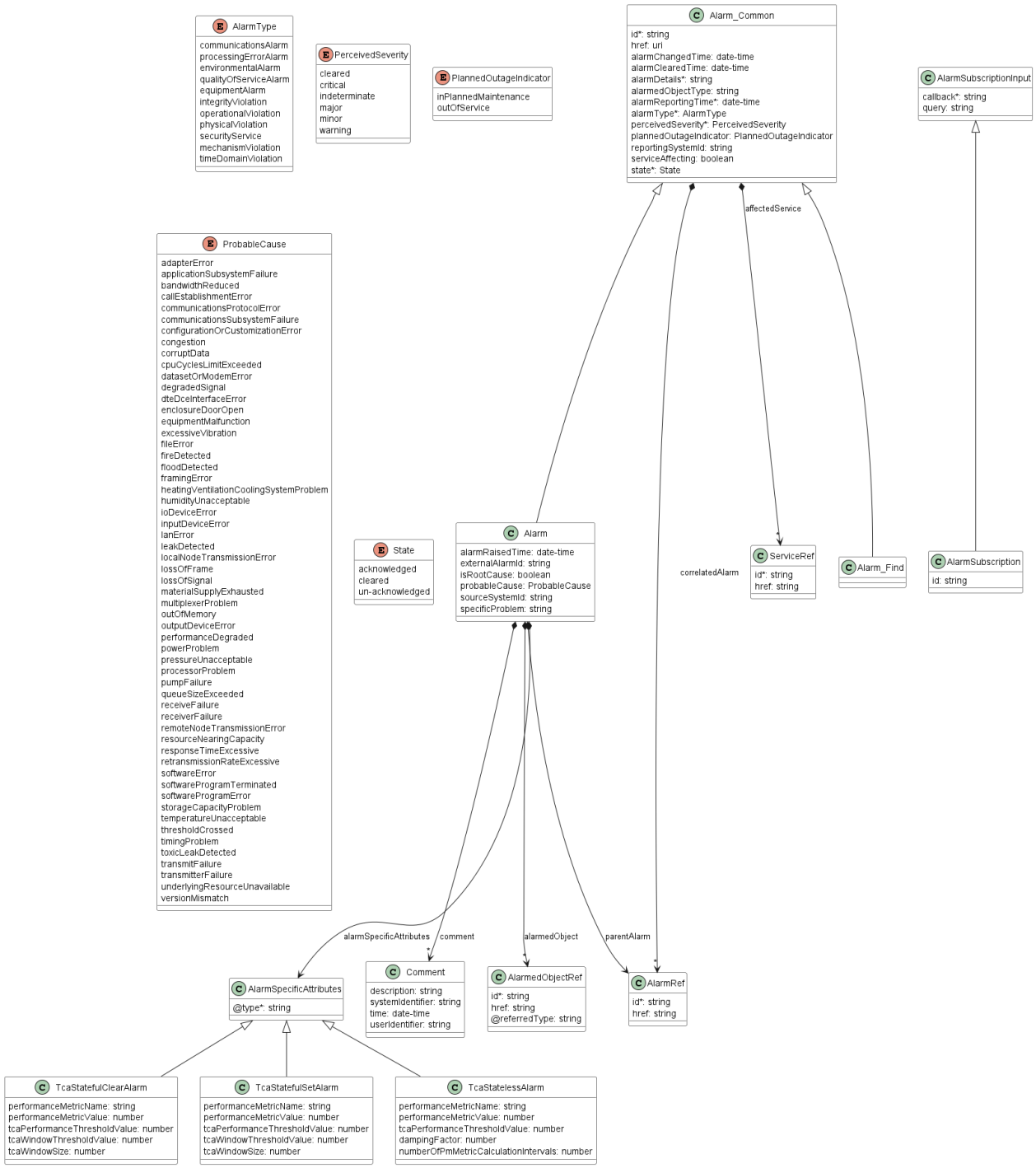

Figure 15. Alarm Data Model

7.2.1 Alarm

7.2.1.1 Type Alarm

Description: A definition of an Alarm.

Inherits from:

- Alarm_Common

Is Root Cause

Name	Type	M/O	Description	MEF 133.1

Name	Type	M/O	Description	MEF 133.1
alarmRaisedTime	date-time *format = date-time*	O	The time that an alarm was raised. This time may differ from the Alarm Reported Time	Alarm Raised Time
alarmedObject	AlarmedObjectRef[]	O	Identifies the Managed Object instance associated with the alarm.	Alarmed Object
comment	Comment[]	O		Comment
externalAlarmId	string	O	An identifier of the alarm in the source system.	External Alarm Identifier
isRootCause	boolean	O	Indicates whether the alarm is a root cause alarm.	
parentAlarm	AlarmRef	O		Parent Alarm
probableCause	string	O	list of alarm probable cause as defined in ITU - T Recommendation X.733 or 3GPP TS 32.111 - 2 Annex B.	Probable Cause
alarmSpecificAttributes	AlarmSpecificAttributes	O		
sourceSystemId	string	O	Source system identity.	Source System Identifier
specificProblem	string	O	Provides more specific information about the alarm.	Specific Problem

7.2.1.2 Type AlarmRef

Description:

Name	Type	M/O	Description	MEF 133.1
id	string	M	unique identifier of the Alarm	Alarm Identifier
href	string	O	hyperlink to the referenced Alarm	

7.2.1.3 Type AlarmSpecificAttributes

Description: AlarmSpecificAttributes is used as an extension for Alarm specific payload. The @type attribute is used as a discriminator.

Name	Type	M/O	Description	MEF 133.1

Name	Type	M/O	Description	MEF 133.1
@type	string	M	The named type must be a subclass of AlarmSpecificAttributes.	

7.2.1.4 enum AlarmType

Description: Categorize the alarm. Values as defined in X.733 8.1.1 or 3GPP TS 32.111.

state	MEF 133 name	Description
communicationsAlarm	Communications Alarm	An alarm of this type is principally associated with the procedures and/or processes required to convey information from one point to another.
processingErrorAlarm	Processing Error Alarm	An alarm of this type is principally associated with a software or processing fault.
environmentalAlarm	Environmental Alarm	An alarm of this type is principally associated with a condition relating to an enclosure in which the equipment resides.
qualityOfServiceAlarm	Quality of Service Alarm	An alarm of this type is principally associated with a degradation in the quality of a service.
equipmentAlarm	Equipment Alarm	An alarm of this type is principally associated with an equipment fault.
communicationsAlarm	Communications Alarm	An alarm of this type is principally associated with the procedures and/or processes required to convey information from one point to another.
processingErrorAlarm	Processing Error Alarm	An alarm of this type is principally associated with a software or processing fault.
environmentalAlarm	Environmental Alarm	An alarm of this type is principally associated with a condition relating to an enclosure in which the equipment resides.
qualityOfServiceAlarm	Quality of Service Alarm	An alarm of this type is principally associated with a degradation in the quality of a service.
equipmentAlarm	Equipment Alarm	An alarm of this type is principally associated with an equipment fault.
integrityViolation	Integrity Violation	Information may have been illegally modified, inserted or deleted
operationalViolation	Operational Violation	The unavailability, malfunction or incorrect invocation of a service.
physicalViolation	Physical Violation	A physical resource has been violated in a way that suggests a security attack.
securityService	Security Service	A security attack has been detected by a security service.
mechanismViolation	Mechanism Violation	A security attack has been detected by a security mechanism.

state	MEF 133 name	Description
timeDomainViolation	Time Domain Violation	An event has occurred at an unexpected or prohibited time.

Value	MEF 133.1
communicationsAlarm	COMMUNICATIONS_ALARM
processingErrorAlarm	PROCESSING_ERROR_ALARM
environmentalAlarm	ENVIRONMENTAL_ALARM
qualityOfServiceAlarm	QUALITY_OF_SERVICE_ALARM
equipmentAlarm	EQUIPMENT_ALARM
integrityViolation	INTEGRITY_VIOLATION
operationalViolation	OPERATIONAL_VIOLATION
physicalViolation	PHYSICAL_VIOLATION
securityService	SECURITY_SERVICE
mechanismViolation	MECHANISM_VIOLATION
timeDomainViolation	TIME_DOMAIN_VIOLATION

7.2.1.5 Type Alarm_Common

Description: A definition of an Alarm.

Name	Type	M/O	Description	MEF 133.1
id	string	M	The identifier of the Alarm.	Alarm Identifier
href	uri *format = uri*	O	Hyperlink reference	
affectedService	ServiceRef[]	O	Affected services	Affected Service
alarmChangedTime	date-time *format = date-time*	O	Indicates the last date and time when the alarm is changed on the alarm owning system. Any change to the alarm whether coming from the alarmed resource is changing this time.	Alarm Changed Time
alarmClearedTime	date-time *format = date-time*	O	Indicates the time (as a date + time) at which the alarm is cleared at the source.	Alarm Cleared Time
alarmDetails	string	M	Contains further information on the Alarm.	Description
alarmedObjectType	string	O	The type (class) of the Managed Object associated with the event e.g. port.	Alarmed Object Type

Name	Type	M/O	Description	MEF 133.1
alarmReportingTime	date-time *format = date-time*	M	Indicates the time (as a date + time) at which the alarm was reported by the owning OSS. It might be different from the alarm-RaisedTime. For instance, if the alarm list is maintained by an EMS, the alarmRaisedtime would be the time the alarm was detected by the NE, while the alarmReportingTime would be the time this alarm was stored in the alarm list of the EMS.	Alarm Reporting Time
correlatedAlarm	AlarmRef[]	O	Correlated Alarms	Correlated Alarm
alarmType	AlarmType	M		Alarm Type
perceivedSeverity	PerceivedSeverity	M		Perceived Severity
plannedOutageIndicator	string	O	Indicates that the Managed Object (related to this alarm) is in in planned maintenance, or out of service	Planned Outage Indicator
reportingSystemId	string	O	Reporting system identity.	Reporting System Identifier
serviceAffecting	boolean	O	Indicates whether the alarm affects service or not.	Service Affecting
state	string	M	Defines the alarm state during its life cycle.	State

7.2.1.6 Type Alarm_Find

Description:

Inherits from:

- Alarm_Common

7.2.1.7 Type AlarmedObjectRef

Description:

Name	Type	M/O	Description	MEF 133.1

Name	Type	M/O	Description	MEF 133.1
id	string	M	unique identifier of the Alarm	Alarm Identifier
href	string	O	hyperlink to the referenced Alarm	
@referredType	string	O	The actual type of the target instance when needed for disambiguation.	

7.2.1.8 Type Comment

Description: A comment entered on the alarm

Name	Type	M/O	Description	MEF 133.1
description	string	O	The text of the comment.	Comment
systemIdentifier	string	O	The system identifier of the system that set the comment.	System Identifier
time	date-time *format = date-time*	O	The time commenting the alarm	Time
userIdentifier	string	O	The user commenting the alarm	

7.2.1.9 `enum` PerceivedSeverity

Description: List of possible severities that can be allocated to an Alarm. The values are consistent with ITU - T Recommendation X.733. Once an alarm has been cleared, its perceived severity is set to 'cleared' and can no longer be set

state	MEF 133 name	Description
`cleared`	Cleared	The Cleared severity level indicates the clearing of one or more previously reported alarms.
`critical`	Critical	The Critical severity level indicates that a service affecting condition has occurred and an immediate corrective action is required
`indeterminate`	Indeterminate	The Indeterminate severity level indicates that the severity level cannot be determined.
`major`	Major	The Major severity level indicates that a service affecting condition has developed and an urgent corrective action is required
`minor`	Minor	The Minor severity level indicates the existence of a non-service affecting fault condition and that corrective action should be taken in order to prevent a more serious (for example, service affecting) fault
`warning`	Warning	The Warning severity level indicates the detection of a potential or impending service affecting fault, before any significant effects have been felt

Value	MEF 133.1
cleared	CLEARED

Value	MEF 133.1
critical	CRITICAL
indeterminate	INDETERMINATE
major	MAJOR
minor	MINOR
warning	WARNING

7.2.1.10 enum PlannedOutageIndicator

Description: Indicates that the Managed Object (related to this alarm) is in in planned maintenance, or out of service

Value	MEF 133.1
inPlannedMaintenance	IN_PLANNED_MAINTENANCE
outOfService	OUT_OF_SERVICE

7.2.1.11 enum ProbableCause

Description: list of alarm probable cause as defined in ITU - T Recommendation X.733 or 3GPP TS 32.111 - 2 Annex B.

Value	MEF 133.1
adapterError	ADAPTER_ERROR
applicationSubsystemFailure	APPLICATION_SUBSYSTEM_FAILURE
bandwidthReduced	BANDWIDTH_REDUCED
callEstablishmentError	CALL_ESTABLISHMENT_ERROR
communicationsProtocolError	COMMUNICATIONS_PROTOCOL_ERROR
communicationsSubsystemFailure	COMMUNICATIONS_SUBSYSTEM_FAILURE
configurationOrCustomizationError	CONFIGURATION_OR_CUSTOMIZATION_ERROR
congestion	CONGESTION
corruptData	CORRUPT_DATA
cpuCyclesLimitExceeded	CPU_CYCLES_LIMIT_EXCEEDED
datasetOrModemError	DATASET_OR_MODEM_ERROR
degradedSignal	DEGRADED_SIGNAL
dteDceInterfaceError	DTE_DCE_INTERFACE_ERROR
enclosureDoorOpen	ENCLOSURE_DOOR_OPEN
equipmentMalfunction	EQUIPMENT_MALFUNCTION
excessiveVibration	EXCESSIVE_VIBRATION
fileError	FILE_ERROR
fireDetected	FIRE_DETECTED
floodDetected	FLOOD_DETECTED
framingError	FRAMING_ERROR

Value	MEF 133.1
heatingVentilationCoolingSystemProblem	HEATING_VENTILATION_COOLING_SYSTEM_PROB
humidityUnacceptable	HUMIDITY_UNACCEPTABLE
ioDeviceError	IO_DEVICE_ERROR
inputDeviceError	INPUT_DEVICE_ERROR
lanError	LAN_ERROR
leakDetected	LEAK_DETECTED
localNodeTransmissionError	LOCAL_NODE_TRANSMISSION_ERROR
lossOfFrame	LOSS_OF_FRAME
lossOfSignal	LOSS_OF_SIGNAL
materialSupplyExhausted	MATERIAL_SUPPLY_EXHAUSTED
multiplexerProblem	MULTIPLEXER_PROBLEM
outOfMemory	OUT_OF_MEMORY
outputDeviceError	OUTPUT_DEVICE_ERROR
performanceDegraded	PERFORMANCE_DEGRADED
powerProblem	POWER_PROBLEM
pressureUnacceptable	PRESSURE_UNACCEPTABLE
processorProblem	PROCESSOR_PROBLEM
pumpFailure	PUMP_FAILURE
queueSizeExceeded	QUEUE_SIZE_EXCEEDED
receiveFailure	RECEIVE_FAILURE
receiverFailure	RECEIVER_FAILURE
remoteNodeTransmissionError	REMOTE_NODE_TRANSMISSION_ERROR
resourceNearingCapacity	RESOURCE_NEARING_CAPACITY
responseTimeExcessive	RESPONSE_TIME_EXCESSIVE
retransmissionRateExcessive	RETRANSMISSION_RATE_EXCESSIVE
softwareError	SOFTWARE_ERROR
softwareProgramTerminated	SOFTWARE_PROGRAM_TERMINATED
softwareProgramError	SOFTWARE_PROGRAM_ERROR
storageCapacityProblem	STORAGE_CAPACITY_PROBLEM
temperatureUnacceptable	TEMPERATURE_UNACCEPTABLE
thresholdCrossed	THRESHOLD_CROSSED
timingProblem	TIMING_PROBLEM
toxicLeakDetected	TOXIC_LEAK_DETECTED
transmitFailure	TRANSMIT_FAILURE
transmitterFailure	TRANSMITTER_FAILURE
underlyingResourceUnavailable	UNDERLYING_RESOURCE_UNAVAILABLE
versionMismatch	VERSION_MISMATCH

7.2.1.12 Type ServiceRef

Description:

Name	Type	M/O	Description	MEF 133.1
id	string	M	unique identifier of the Service	
href	string	O	hyperlink to the referenced Alarm	

7.2.1.13 enum State

Description: Defines the alarm state during its life cycle.

Value	MEF 133.1
acknowledged	ACKNOWLEDGED
cleared	CLEARED
un-acknowledged	UN-ACKNOWLEDGED

7.2.2 TCA Alarm

7.2.2.1 Type TcaStatefulClearAlarm

Description: Threshold Crossing Alert Alarm Schema.

Inherits from:

- AlarmSpecificAttributes

Name	Type	M/O	Description	MEF 133.1
performanceMetricName	string	O	Human readable text for Performance Metric for which the TCA Function was configured..	Performance Metric Name
performanceMetricValue	number	O	The PM Metric Value for the PM Metric Calculation	PM Metric Value
tcaPerformanceThresholdValue	number	O	The configured TCA Performance Threshold Value for the Performance Metric	TCA Performance Threshold Value
tcaWindowThresholdValue	number	O	The configured TCA Performance Threshold Value for the Performance Metric	SET - TCA Window Threshold Value
tcaWindowSize	number	O	The number of PM Metric Calculation Intervals included in the sliding window	TCA Window Size Value

7.2.2.2 Type TcaStatefulSetAlarm

Description: Threshold Crossing Alert Alarm Schema.

Inherits from:

- AlarmSpecificAttributes

Name	Type	M/O	Description	MEF 133.1
performanceMetricName	string	M	Human readable text for Performance Metric for which the TCA Function was configured..	Performance Metric Name
performanceMetricValue	number	M	The PM Metric Value for the PM Metric Calculation	PM Metric Value
tcaPerformanceThresholdValue	number	M	The configured TCA Performance Threshold Value for the Performance Metric	TCA Performance Threshold Value
tcaWindowThresholdValue	number	M	The configured TCA Performance Threshold Value for the Performance Metric	CLEAR - TCA Window Threshold Value
tcaWindowSize	number	M	The number of PM Metric Calculation Intervals included in the sliding window	TCA Window Size Value

7.2.2.3 Type TcaStatelessAlarm

Description: Threshold Crossing Alert Alarm Schema.

Inherits from:

- AlarmSpecificAttributes

Name	Type	M/O	Description	MEF 133.1
performanceMetricName	string	M	Human readable text for Performance Metric for which the TCA Function was configured..	Performance Metric Name
performanceMetricValue	number	M	The PM Metric Value for the PM Metric Calculation	Performance Metric Value

Name	Type	M/O	Description	MEF 133.1
tcaPerformanceThresholdValue	number	M	The configured TCA Performance Threshold Value for the Performance Metric	TCA Performance Threshold Value
dampingFactor	number	O	The value that identifies the number of PM Metric Calculation Intervals included in the Damping Factor process.	Damping Factor
numberOfPmMetricCalculationIntervals	number	M	The number of PM Metric Calculation Intervals in the hopping window in which the PM Metric Value ? the TCA Performance Threshold Value	Number of PM Metric Calculation Intervals

7.2.3. Notification registration

Notification registration and management are done through `/hub` API endpoint. The below sections describe data models related to this endpoint.

7.2.3.1. Type AlarmSubscriptionInput

Description: This class is used to register for Notifications.

Name	Type	M/O	Description
callback	string	M	This callback value must be set to *host* property from Alarm Notificatio API (alarmNotification.api.yaml). This property is appended with the base patl and notification resource path specified in that API to construct an URL t which notification is sent. E.g. for 'callback' "https://buyer.co/listenerEndpoint", the alarm event notification will be sent to `https://buyer.co/listenerEndpoint/mefApi/legato/alarmNotification/v1/listener/
query	string	O	This attribute is used to define which type of events to register to. Example 'query':'eventType = createAlarm'.

7.2.3.2. Type AlarmSubscription

Description: This resource is used to respond to notification subscriptions.

Inherits from:

- AlarmSubscriptionInput

Name	Type	M/O	Description	MEF 133.1
id	string	O	An identifier of this Event Subscription assigned when a resource is created.	

8. References

- JSON Schema draft 7, JSON Schema: A Media Type for Describing JSON Documents and associated documents, by Austin Wright and Henry Andrews, March 2018. Copyright © 2018 IETF Trust and the persons identified as the document authors. All rights reserved.
- [MEF55.1] MEF 55.1, Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, February 2021
- [MEF128.1] MEF 128.1, LSO API Security Profile, April 2024
- MEFW133.1 Allegro, Interlude and Legato Fault Management and Alarm API BR&UC, January 2025
- [OAS-v3] Open API 3.0, February 2020
- [REST] Chapter 5: Representational State Transfer (REST) Fielding, Roy Thomas, Architectural Styles and the Design of Network-based Software Architectures (Ph.D.).
- [RFC2119] RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, by S. Bradner, March 1997
- [RFC3986] RFC 3986 Uniform Resource Identifier (URI): Generic Syntax, January 2005
- [RFC8174] RFC 8174, Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, by B. Leiba, May 2017, Copyright © 2017 IETF Trust and the persons identified as the document authors. All rights reserved.
- [TMF630] TMF 630 TMF630 REST API Design Guidelines 4.2.0
- [TMF642] TMF 642, TMF642 Alarm User Guide v5.0.0, September 2023