MPLIFY

Mplify Standard
Mplify 146

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

February 2026

Table of Contents

List of Contributing Members

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

Member
Amartus

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 Mplify 133.1 [Mplify133.1]. The requirements and use cases are the same for all IRPs. This standard consists of this document and complementary API definitions for Alarms and Threshold Crossing Alerts.

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

commit id: d0a8f15a6dfe451d72a92fa5a897696af8a2bb53

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

MEF-LSO-Interlude-SDK

commit id: 59fc0f8445fa59d6e8b9ad6336fdad435d2258a1

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

MEF-LSO-Legato-SDK

commit id: 1de1dd6074358d6378e9eef20b24e8f937ab6033

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

The Threshold Crossing Alerts (Alarms) 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 Mplify 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:

- Mplify 133.1 *Allegro, Interlude and Legato Fault Management and Performance Monitoring BR&UC* [Mplify133.1]
- 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.	Mplify 133.1
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	Mplify 133.1
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.	Mplify 133.1

Table 2. Terminology

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	Mplify 133.1
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 Threshold Crossing Alerts (Alarms) 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 Threshold Crossing Alert (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 Management 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* [MEF 55.1]. The LSO Reference Architecture is shown in Figure 1 with the three IRPs highlighted.

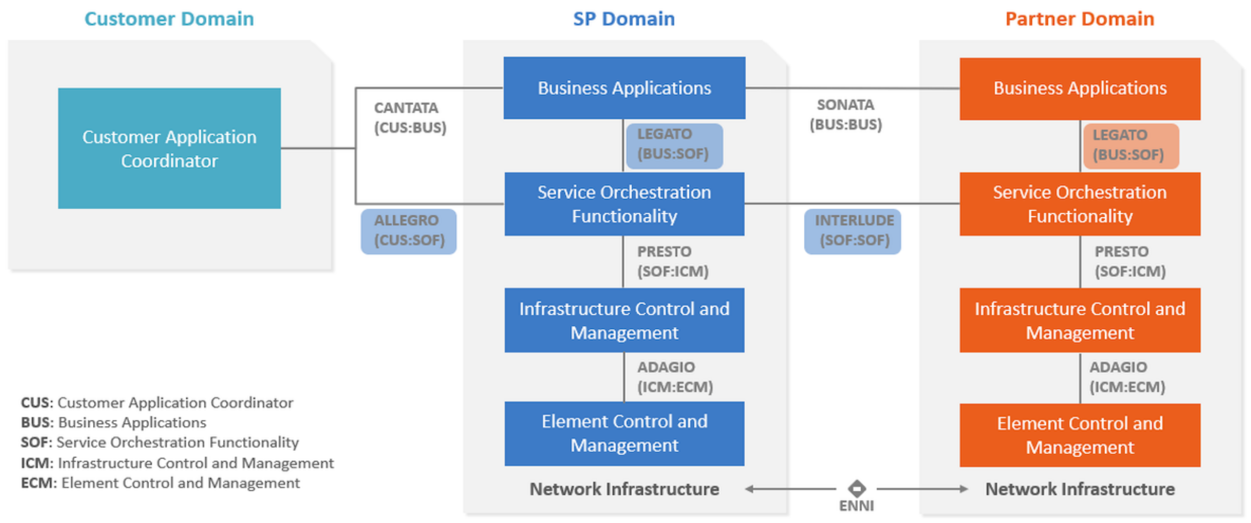

Customer Domain
SP Domain
Partner Domain
Customer Application Coordinator
Business Applications
Service Orchestration Functionality
Infrastructure Control and Management
Element Control and Management
Network Infrastructure
CANTATA (CUS:BUS)
ALLEGRO (CUS:SOF)
LEGATO (BUS:SOF)
PRESTO (SOF:ICM)
ADAGIO (ICM:ECM)
SONATA (BUS:BUS)
INTERLUDE (SOF:SOF)
ENNI
CUS: Customer Application Coordinator
BUS: Business Applications
SOF: Service Orchestration Functionality
ICM: Infrastructure Control and Management
ECM: Element Control and Management

Figure 1. The LSO Reference Architecture

4.1. Description

The scope of this API and Developer Guide covers

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

The business requirements and use cases for Threshold Crossing Alerts (Alarms) API are defined in Mplify 133.1 Allegro, Interlude and Legato Fault Management and Performance Monitoring BR&UC Mplify133.1.

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. `Alarm`).
- 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 Threshold Crossing Alerts (Alarms) related requirements and use cases that are defined in [Mplify133.1]. 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 (Mplify service specification data model)

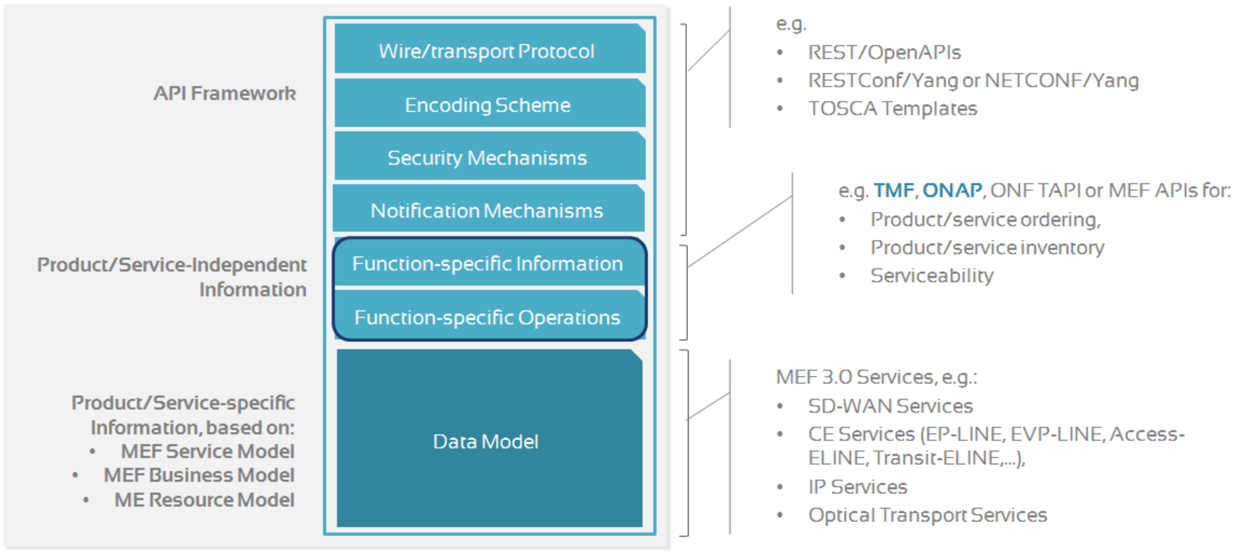

API Framework
Product/Service-Independent Information
Product/Service-specific Information, based on:
• MEF Service Model
• MEF Business Model
• ME Resource Model
Wire/transport Protocol
Encoding Scheme
Security Mechanisms
Notification Mechanisms
Function-specific Information
Function-specific Operations
Data Model
e.g.
• REST/OpenAPIs
• RESTConf/Yang or NETCONF/Yang
• TOSCA Templates
e.g. TMF, ONAP, ONF TAPI or MEF APIs for:
• Product/service ordering,
• Product/service inventory
• Serviceability
MEF 3.0 Services, e.g.:
• SD-WAN Services
• CE Services (EP-LINE, EVP-LINE, Access-ELINE, Transit-ELINE,...),
• IP Services
• Optical Transport Services

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 describes the Alarm Model and operations that enable the management of alarms aligned with either Mplify or custom alarm specifications.

This Developer Guide does not define Mplify alarm specifications, but it can be used in conjunction with any alarm specifications defined by or compliant with Mplify.

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

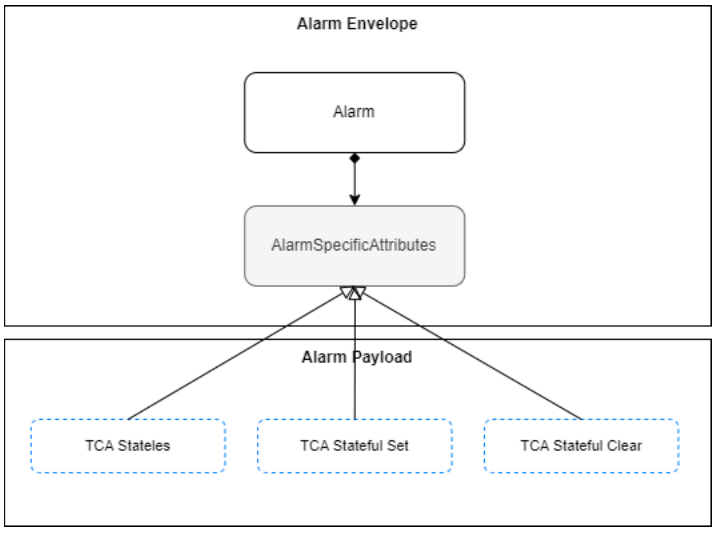


Figure 3. Alarm specification for Allegro, Interlude, Legato

4.5. High-Level Flow

The Threshold Crossing Alerts (Alarms) API allows the Buyer to retrieve alarms and receive notifications when an alarm is created, deleted, acknowledged, unacknowledged, commented or cleared. This enables timely detection and resolution of faults related to alarms provided by the Seller.

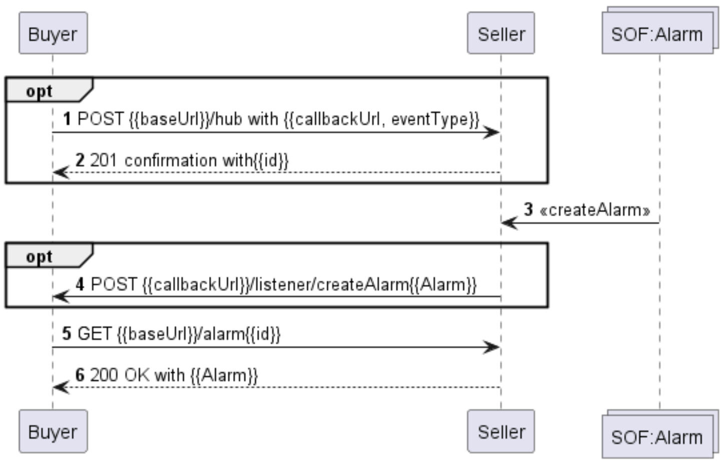

Buyer
Seller
SOF:Alarm
opt
1 POST {{baseUrl}}/hub with {{callbackUrl, eventType}}
2 201 confirmation with{{id}}
3 «createAlarm»
opt
4 POST {{callbackUrl}}/listener/createAlarm{{Alarm}}
5 GET {{baseUrl}}/alarm{{id}}
6 200 OK with {{Alarm}}
Buyer
Seller
SOF:Alarm

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 condition 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.

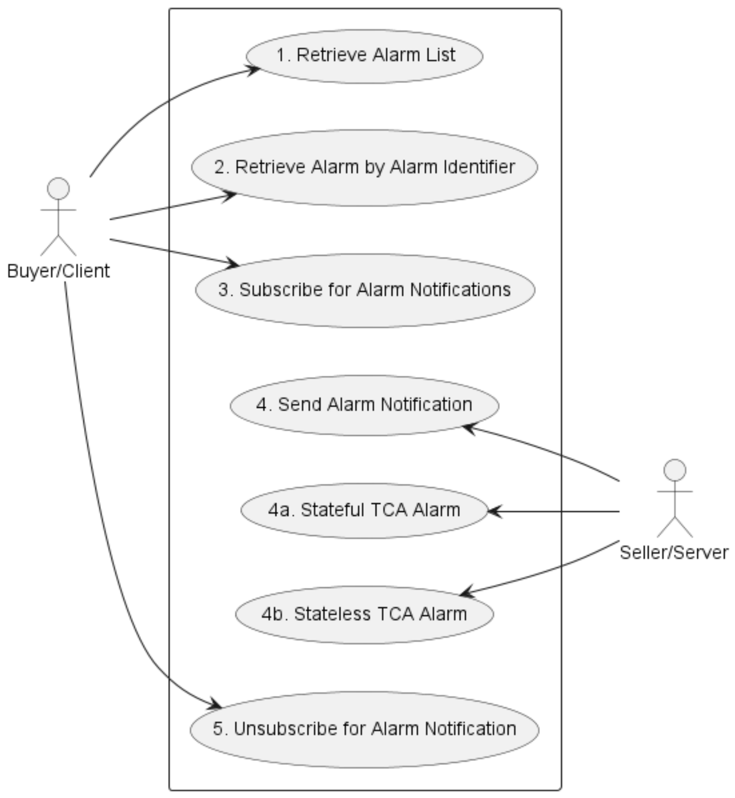

Buyer/Client
Seller/Server
1. Retrieve Alarm List
2. Retrieve Alarm by Alarm Identifier
3. Subscribe for Alarm Notifications
4. Send Alarm Notification
4a. Stateful TCA Alarm
4b. Stateless TCA Alarm
5. Unsubscribe for Alarm Notification

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/v3/`

Base URL for Interlude:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/interlude/alarmManagement/v3/`

Base URL for Legato:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/legato/alarmManagement/v3/`

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	Mplify 133.1 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
POST /hub	The Buyer/Client requests to subscribe to the Alarm Notifications.	UC 51: Subscribe to Alarms
GET /hub/{{id}}	The Buyer/Client retrieves a specific EventSubscription from the Seller/Server, that matches the *id* value provided as *path* parameter.	
DELETE /hub/{{id}}	The Buyer/Client requests to unsubscribe from the Alarm Notifications.	UC 52: Un-Subscribe from Alarms

Table 4. Seller/Server mandatory API endpoints

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

5.2.2. Buyer/Client side API Endpoints

Base URL for Allegro:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/allegro/alarmNotification/v3/`

Base URL for Interlude:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/interlude/alarmNotification/v3/`

Base URL for Legato:

`https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/legato/alarmNotification/v3/`

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	Mplify 133.1 Use Case Mapping
POST /listener/alarmAttributeValueChangeEvent	A request initiated by the Seller/Server to notify Buyer/Client on Alarm instance attribute value change.	UC 48: Send Alarm Notification

API Endpoint	Description	Mplify 133.1 Use Case Mapping
`POST /listener/alarmCreateEvent`	A request initiated by the Seller/Server to notify Buyer/Client on `Alarm` instance create.	UC 48: Send Alarm Notification
`POST /listener/alarmDeleteEvent`	A request initiated by the Seller/Server to notify Buyer/Client on `Alarm` instance delete.	UC 48: Send Alarm Notification
`POST /listener/alarmStateChangeEvent`	A request initiated by the Seller/Server to notify Buyer/Client on `Alarm` instance state change.	UC 48: Send Alarm Notification

Table 5. Buyer/Client API endpoints

[R2] The Buyer/Client **MUST** support API endpoints listed in Table 5.

[R3] The Buyer/Client **MUST** register to receive Alarm Notifications.

[R4] The Seller/Server **MUST** support sending notifications to API endpoints listed in Table 5 to the registered Buyer/Client.

5.3. Integration of Alarm Specification into Threshold Crossing Alerts (Alarms) API

The Threshold Crossing Alerts (Alarms) API data model discussed in this document is a generic envelope that enables the management of relevant alarm objects. The API data model itself does not provide explicit definitions or prescribe the structure for different types of alarms. However, it offers flexible extensibility to accommodate the configuration of alarm-specific objectives, allowing for customization and adaptation to various requirements. The Threshold Crossing Alerts (Alarms) API schema is defined using JSON Schema Draft 7 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 Mplify standard service schemas is in URN format and is assigned by Mplify. This identifier is provided as root schema `$id`. Use of non-Mplify 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 Threshold Crossing Alerts (Alarms) 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 Threshold Crossing Alerts (Alarms) 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 Mplify 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:service:tca-stateful-set-alarm:v1.0.0:all: '#/components/schemas/TcaStatefulSetAlarm'
      urn:mef:lso:spec:service:tca-stateless-alarm:v1.0.0:all: '#/components/schemas/TcaStatelessAlarm'
      urn:mef:lso:spec:service:tca-stateful-clear-alarm:v1.0.0: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:service:tca-stateful-set-alarm:v1.0.0:all
        - urn:mef:lso:spec:service:tca-stateless-alarm:v1.0.0:all
        - urn:mef:lso:spec:service:tca-stateful-clear-alarm:v1.0.0: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.

[R5] `AlarmSpecificAttributes` type is extension points that **MUST** be used to integrate alarm specific properties into a request/response payload.

[R6] The `@type` property of `AlarmSpecificAttributes` **MUST** be used to specify the type of the extending entity.

[R7] Attributes specified in the payload must conform to the alarm definition specified in the `@type` property.

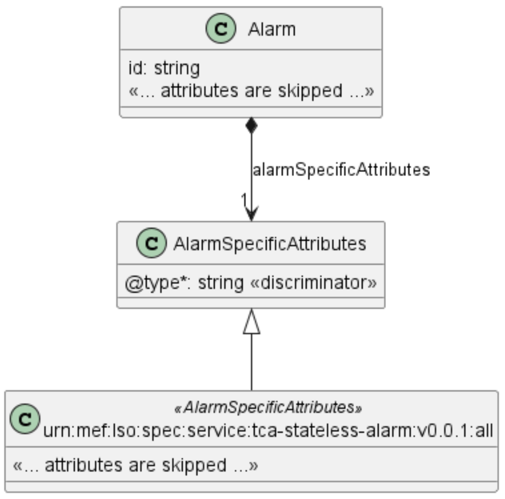

Alarm
id: string
«... attributes are skipped ...»
alarmSpecificAttributes
1
AlarmSpecificAttributes
@type*: string «discriminator»
«AlarmSpecificAttributes»
urn:mef:lso:spec:service:tca-stateless-alarm:v0.0.1:all
«... attributes are skipped ...»

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

Figure 6 presents two Mplify Threshold Crossing Alerts (Alarms) API schema that represent alarm configuration for TCA Stateless Alarm. When this schema is used, the `@type` of `AlarmSpecificAttributes` takes`"urn:mef:lso:spec:legato:tca-stateless-alarm:v0.0.1:all"` value to indicate which alarm specification should be used to interpret asset 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 Threshold Crossing Alerts (Alarms) API endpoints is defined using:

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

[R8] Implementations **MUST** use payloads that conform to these definitions. The Buyer and the Seller **MUST NOT** use any operation, entity or attribute that is not explicitly defined or allowed by this standard.

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* [MEF 128.1].

6. API Interactions and Flows

This section provides a detailed insight into the API functionality, use cases, and flows. It starts with Table 6 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	Mplify 133.1 Use Case #
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.	49
2	Retrieve Alarm by Alarm Identifier	The Buyer/Client requests detailed information about a single Alarm based on the Alarm Identifier.	50
3	Subscribe to Alarm Notifications	The Buyer/Client requests to subscribe to Alarm Notifications.	51
4	Send Alarm Notification	A Seller/Server sends an Alarm to the Buyer/Client based on an event that has occurred.	48
4a	Stateful TCA Alarm	A Stateful TCA Alarm is initiated by the Seller/Server to a subscribed Client.	53
4b	Stateless TCA Alarm	A Stateless TCA lifecycle alarm is initiated by the Seller/Server to a subscribed Client.	54
5	Unsubscribe for Alarm Notifications	The Buyer/Client requests to unsubscribe to Alarm and/or Test Job Notifications.	52

Table 6. Use cases description

6.1. Use Case 1: Retrieve List of Alarms

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

[R9] The Buyer/Client Retrieve List of Alarms request **MUST** contain none or more of the following attributes as filter criteria: [Mplify133.1 R138]

- `id`
- `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/v3/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.

[R10] The Seller/Server MUST support the Retrieve Alarm List Use Case. [Mplify133.1 R139]

[R11] The Buyer/Client MUST support the Retrieve Alarm List Use Case. [Mplify133.1 R140]

[R12] The Seller/Server's response to the Buyer's/Client's Retrieve List of Alarms **MUST** include all attributes that were provided as filter criteria. In addition, the following attributes **MUST** be also included in the response. [Mplify133.1 R141]

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

[O1] The Seller response MAY contain any of the remaining Alarm attributes [Mplify133.1 O25]

[R13] In case no items matching the criteria are found, the Seller/Server **MUST** return a valid response with an empty list.

[R14] If the request is unsuccessful, the Seller/Server **MUST** return an error with explanation to the Buyer/Client.

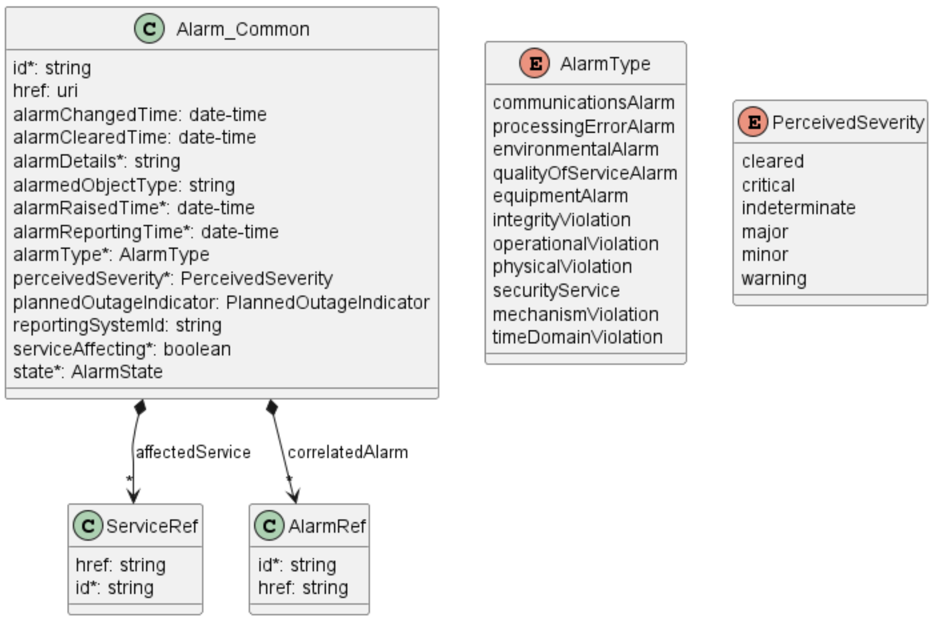


Figure 7. Use Case 1: Retrieve Alarm List - Model

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.

[R15] The Seller/Server MUST support the Retrieve Alarm by Identifier Use Case [Mplify133.1 R142]

[R16] The Buyer/Client MUST support the Retrieve Alarm by Identifier Use Case [Mplify133.1 R143]

[R17] The Buyer's Retrieve Alarm by Alarm Identifier **MUST** include the Alarm Identifier. [Mplify133.1 R144]

[R18] The Buyer's Retrieve Alarm by Alarm Identifier **MUST NOT** include other attributes. [Mplify133.1 R145]

[R19] If the request is successful, the Seller's response to a Retrieve Alarm by Alarm Identifier request MUST include all attributes

[R20] If the request is unsuccessful, the Seller/Server **MUST** return an error with explanation to the Buyer/Client.

[R21] In case `id` does not allow finding a `Alarm` in Seller/Server's system, an error response `Error404` **MUST** be returned.

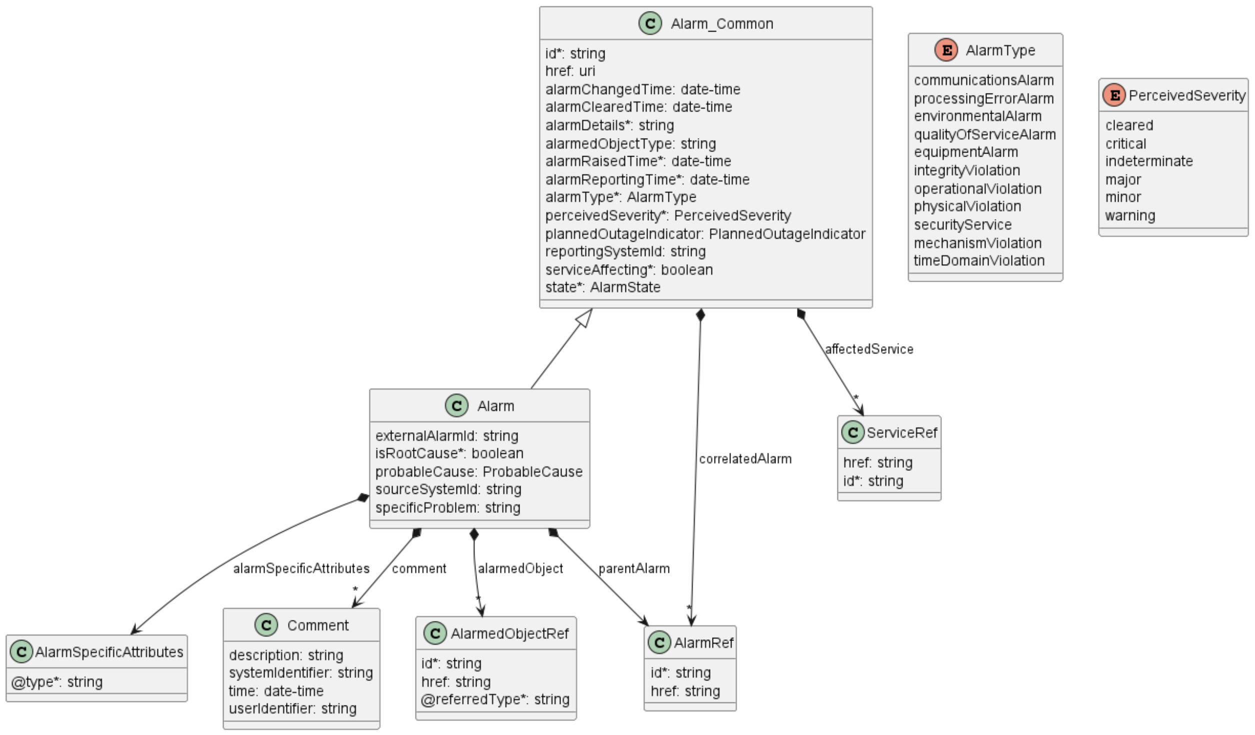
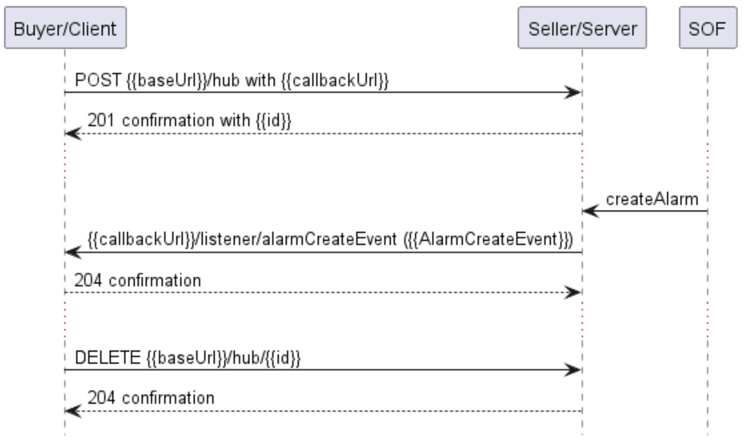


Figure 8. Use Case 2: Retrieve Alarm by Alarm Identifier - Model

6.3. Use Case 3: Subscribe to Alarms

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



Buyer/Client
Seller/Server
SOF
POST {{baseUrl}}/hub with {{callbackUrl}}
201 confirmation with {{id}}
createAlarm
{{callbackUrl}}/listener/alarmCreateEvent ({{AlarmCreateEvent}})
204 confirmation
DELETE {{baseUrl}}/hub/{{id}}
204 confirmation

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 only two attributes:

- `callback` - mandatory, to provide the callback address the alarm notifications will be sent to,
- `query`- optional, to provide the required types of event.

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

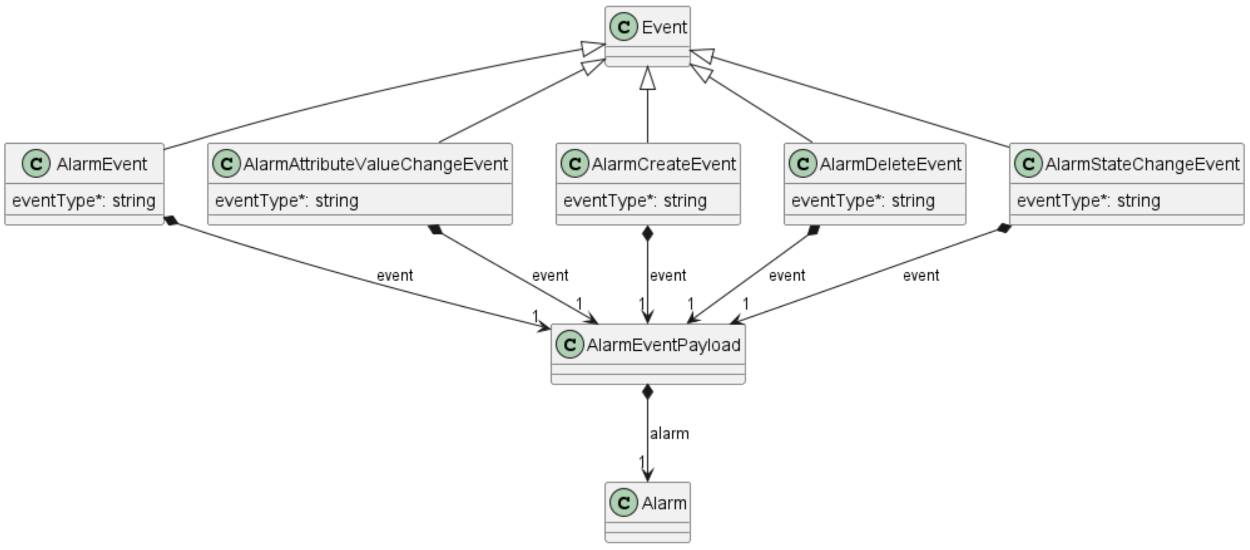


Figure 10. Threshold Crossing Alerts (Alarms) API Notification Data Model

By using a request in the following snippet,

```
{
  "callback": "https://client.mef.com/listenerEndpoint"
}
```

the Buyer/Client subscribes for alarms of all types of events. Those are:

- `alarmAttributeValueChangeEvent`
- `alarmCreateEvent`
- `alarmDeleteEvent`
- `alarmStateChangeEvent`

[R22] The Seller/Server **MUST** support subscription to Alarms Use Case.

[R23] The Seller/Server **MUST** support unsubscribing from Alarms Use Case.

[R24] The Buyer/Client's Subscribe to Alarms request **MUST** include: [Mplify133.1 R146]

- `callback`

If the Buyer/Client wishes to receive only notifications of a certain type, a `query` parameter must be added to the request:

```
{
  "callback": "https://client.mef.com/listenerEndpoint",
  "query": "alarmCreateEvent"
}
```

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 = alarmCreateEvent",
  "callback": "https://client.mef.com/listenerEndpoint"
}
```

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

- `https://client.mef.com/listenerEndpoint/mefApi/legato/alarmNotification/v3/listener/alarmCreateEvent`

[R25] If successful, the Seller/Server response **MUST** indicate success and include the Register Notification Identifier and echo back all Buyer/Client provided attributes.

[R26] If successful, the Seller/Server **MUST** begin sending the alarms to the Buyer/Client.

[R26] The Seller/Server **MUST NOT** send alarms if the Buyer/Client has not registered for them.

[R28] If unsuccessful, the Seller/Server **MUST NOT** return a Register Notification Identifier.

[R29] If the Seller/Server experiences any errors, they **MUST** return an error indication to the Buyer/Client.

If the Buyer/Client wishes to subscribe to more than one different types of events, there are two possible syntax variants [TMF630]:

```
eventType=alarmCreateEvent,alarmStateChangeEvent
```

or

```
eventType=alarmCreateEvent&eventType=alarmStateChangeEvent
```

6.4. Use Case 4: Send Alarm Notification

Alarms are used to asynchronously inform the Buyer/Client about detected faults or abnormal conditions. The Seller/Server can send the following types of alarm notifications to the Buyer/Client:

- `alarmAttributeValueChangeEvent`
- `alarmCreateEvent`
- `alarmDeleteEvent`
- `alarmStateChangeEvent`

The Figure 11 shows all entities involved in the Send Alarm Notification use cases.

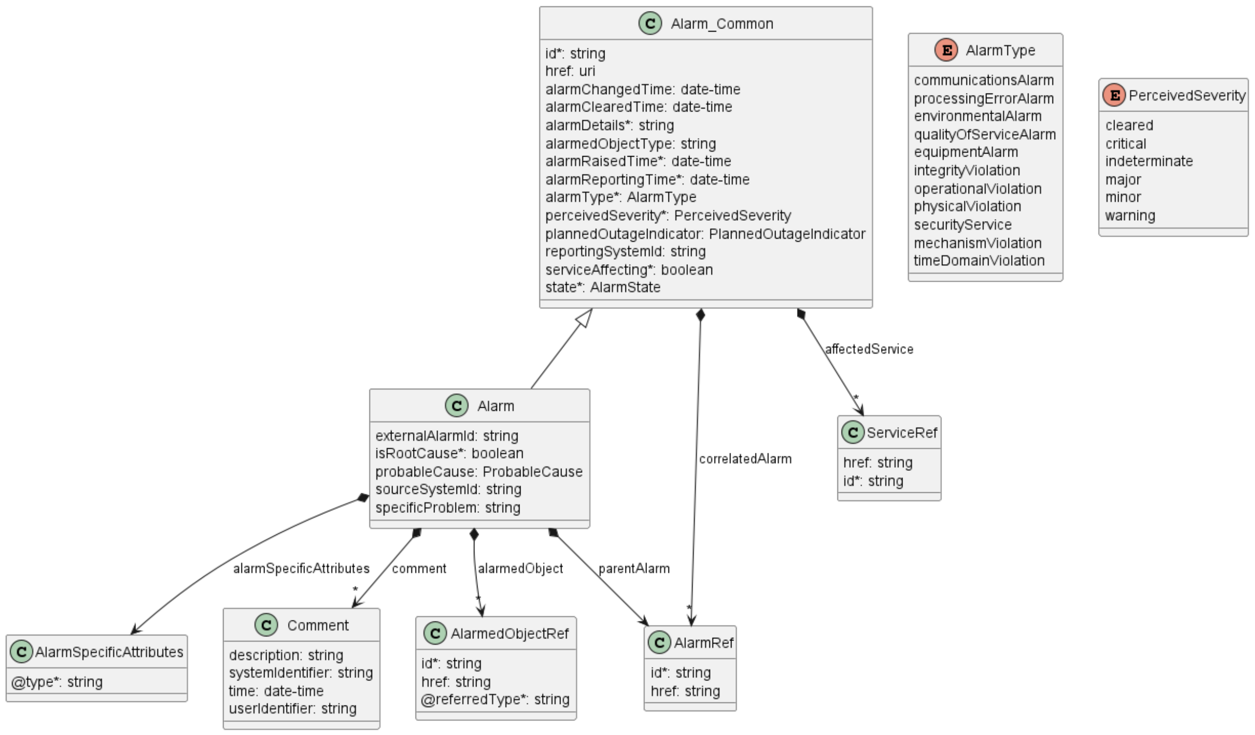


Figure 11. Use Case 4: Alarm - Model

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-20250601-001",
  "href": "https://serverRoot/mefApi/legato/alarmManagement/v3/alarm/alarm-20250601-001",
  "affectedService": [
    {
      "href": "https://serverRoot/mefApi/legato/service/service-123",
      "id": "service-123"
    }
  ],
  "alarmChangedTime": "2025-06-01T14:52:57.382Z",
  "alarmClearedTime": "2025-06-01T14:52:57.382Z",
  "alarmDetails": "TCA clear: Latency has dropped below the defined threshold",
  "alarmedObjectType": "ServiceEndpoint",
  "alarmRaisedTime": "2025-06-01T14:50:00.000Z",
  "alarmReportingTime": "2025-06-01T14:52:57.382Z",
  "correlatedAlarm": [
    {
      "id": "alarm-20250601-000",
      "href": "https://serverRoot/mefApi/legato/alarmManagement/v3/alarm/alarm-20250601-000"
    }
  ],
  "alarmType": "qualityOfServiceAlarm",
  "perceivedSeverity": "cleared",
  "plannedOutageIndicator": "inPlannedMaintenance",
  "reportingSystemId": "monitoring-system-01",
  "serviceAffecting": false,
  "state": "acknowledged",
  "alarmedObject": [
    {
      "id": "endpoint-456",
      "href": "https://serverRoot/objects/endpoint-456",
      "@referredType": "ServiceEndpoint"
    }
  ],
  "comment": [
    {
      "description": "Cleared automatically by system upon recovery",
      "systemIdentifier": "auto-clear-daemon",
      "time": "2025-06-01T14:53:00.000Z",
      "userIdentifier": "system"
    }
  ],
  "externalAlarmId": "ext-789-clear",
  "isRootCause": true,
  "parentAlarm": {
    "id": "alarm-20250601-000",
    "href": "https://serverRoot/mefApi/legato/alarmManagement/v3/alarm/alarm-20250601-000"
  },
  "probableCause": "adapterError",
  "alarmSpecificAttributes": {
    "@type": "urn:mef:lso:spec:service:tca-stateful-clear-alarm:v0.0.1:all",
    "performanceMetricName": "oneWayFrameDelay",
    "performanceMetricValue": 7.2,
    "tcaPerformanceThresholdValue": 10.0,
    "tcaWindowThresholdValue": 9.0,
    "tcaWindowSize": 5
  },
  "sourceSystemId": "probe-east-01",
  "specificProblem": "Latency spike resolved below threshold"
}
```

[R30] The Seller/Server **MUST** include the following Alarm attributes: [Mplify133.1 R137]

- `id`
- `alarmDetails`
- `alarmedObject`
- `alarmRaisedTime`
- `alarmReportingTime`
- `alarmType`
- `perceivedSeverity`
- `serviceAffecting`
- `state`
- `isRootCause`

[R31] The Seller/Server **MUST** send Alarms to the Buyer/Client that has registered for them.

[R32] 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

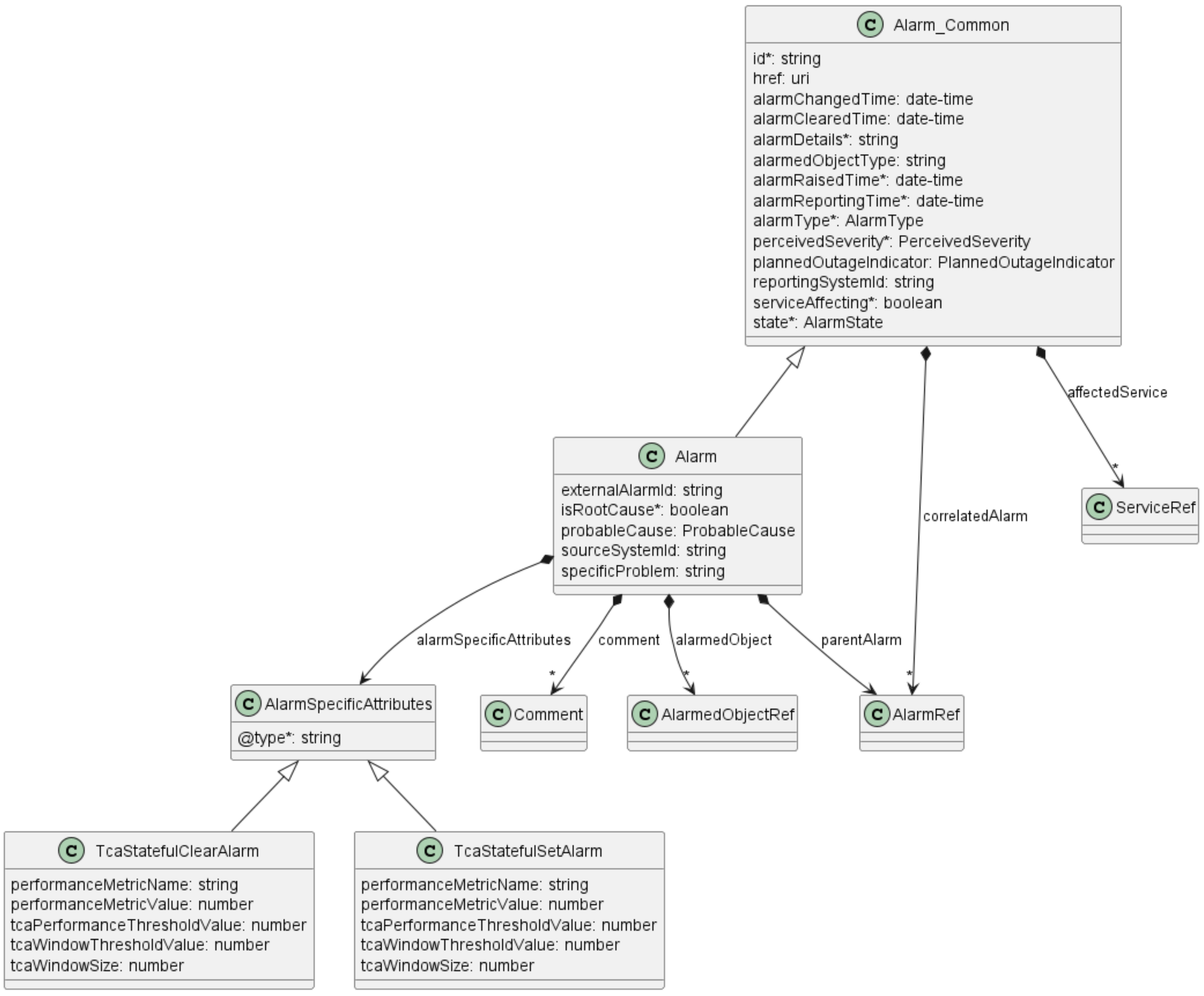

Alarm_Common
id*: string
href: uri
alarmChangedTime: date-time
alarmClearedTime: date-time
alarmDetails*: string
alarmedObjectType: string
alarmRaisedTime*: date-time
alarmReportingTime*: date-time
alarmType*: AlarmType
perceivedSeverity*: PerceivedSeverity
plannedOutageIndicator: PlannedOutageIndicator
reportingSystemId: string
serviceAffecting*: boolean
state*: AlarmState
affectedService
ServiceRef
correlatedAlarm
Alarm
externalAlarmId: string
isRootCause*: boolean
probableCause: ProbableCause
sourceSystemId: string
specificProblem: string
alarmSpecificAttributes
comment
alarmedObject
parentAlarm
AlarmSpecificAttributes
@type*: string
Comment
AlarmedObjectRef
AlarmRef
TcaStatefulClearAlarm
performanceMetricName: string
performanceMetricValue: number
tcaPerformanceThresholdValue: number
tcaWindowThresholdValue: number
tcaWindowSize: number
TcaStatefulSetAlarm
performanceMetricName: string
performanceMetricValue: number
tcaPerformanceThresholdValue: number
tcaWindowThresholdValue: number
tcaWindowSize: number

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

[R33] When sending an alarm for a TCA Reporting Type of Stateful, the Seller/Server **MUST** include `alarmType=qualityOfServiceAlarm` and the following mandatory attributes: [Mplify133.1 R147]

- `performanceMetricName`
- `performanceMetricValue`
- `tcaPerformanceThresholdValue`
- `tcaWindowThresholdValue`
- `tcaWindowSize`

6.4.2 Use Case 4b: Stateless TCA Alarm

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

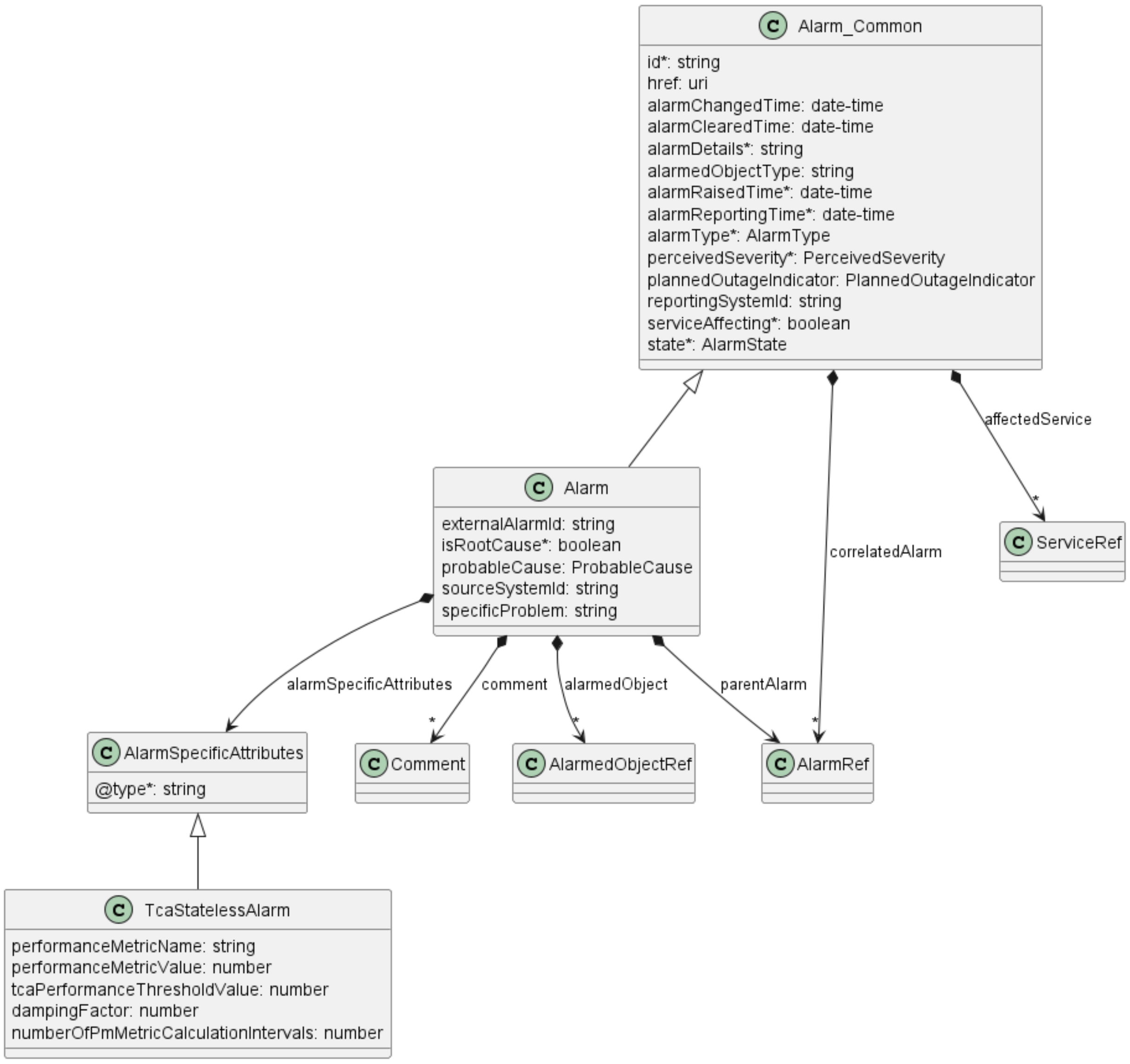

Alarm_Common
id*: string
href: uri
alarmChangedTime: date-time
alarmClearedTime: date-time
alarmDetails*: string
alarmedObjectType: string
alarmRaisedTime*: date-time
alarmReportingTime*: date-time
alarmType*: AlarmType
perceivedSeverity*: PerceivedSeverity
plannedOutageIndicator: PlannedOutageIndicator
reportingSystemId: string
serviceAffecting*: boolean
state*: AlarmState
affectedService
ServiceRef
correlatedAlarm
Alarm
externalAlarmId: string
isRootCause*: boolean
probableCause: ProbableCause
sourceSystemId: string
specificProblem: string
alarmSpecificAttributes
comment
alarmedObject
parentAlarm
AlarmSpecificAttributes
@type*: string
Comment
AlarmedObjectRef
AlarmRef
TcaStatelessAlarm
performanceMetricName: string
performanceMetricValue: number
tcaPerformanceThresholdValue: number
dampingFactor: number
numberOfPmMetricCalculationIntervals: number

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

[R34] When sending an alarm for a TCA Reporting Type of Stateless, the Seller/Server **MUST** include `alarmType=qualityOfServiceAlarm` and the following mandatory attributes: [Mplify133.1 R150]

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

[R35] If the Damping Factor is included in the TCA Profile, the TCA Alarm **MUST** include the `dampingFactor` attribute. [Mplify133.1 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.

[R36] If successful, the Seller/Server response **MUST** indicate success

[R37] If successful, the Seller/Server **MUST** stop sending the appropriate alarms to the Buyer/Client.

[R38] If unsuccessful, the Seller/Server **MUST NOT** stop sending the appropriate alarms to the Buyer/Client.

[R39] 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`.

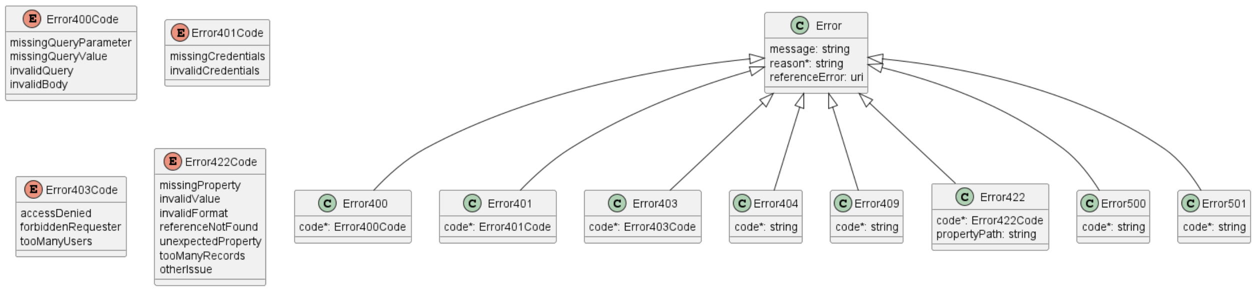


Figure 14. Data model types to represent an erroneous response

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

Name	Type	Description
code*	Error400Code	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

7.1.1.3. `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	One of the following error codes: - missingCredentials: No credentials provided. - invalidCredentials: Provided credentials are invalid or expired

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	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

7.1.1.7. `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 Error409

Description: Conflict (https://datatracker.ietf.org/doc/html/rfc7231#section-6.5.8)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - conflict: The client has provided a value whose semantics are not appropriate for the property.

7.1.1.10. 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

Name	Type	Description
code*	Error422Code	One of the following error codes: - missingProperty: The property the Seller has 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 Seller system - unexpectedProperty: Additional property, not expected by the Seller has been provided - tooManyRecords: the number of records to be provided in the response exceeds the Seller's threshold. - otherIssue: Other problem was identified (detailed information provided in a reason)
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).

7.1.1.11. `enum` Error422Code

Description: One of the following error codes:

- missingProperty: The property the Seller has 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 Seller system
- unexpectedProperty: Additional property, not expected by the Seller has been provided
- tooManyRecords: the number of records to be provided in the response exceeds the Seller's threshold.
- otherIssue: Other problem was identified (detailed information provided in a reason)

7.1.1.12. 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.

7.1.1.13. 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 Mplify 133.1 specification are discussed later in this section.

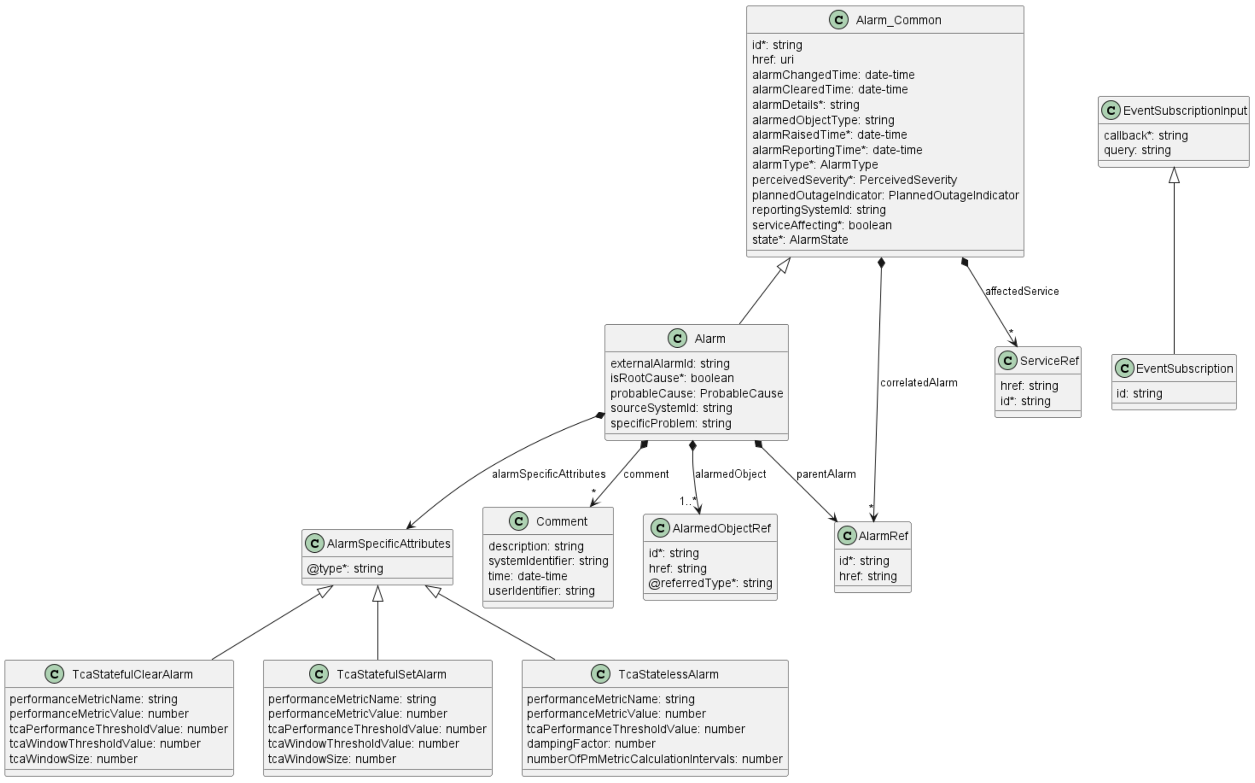


Figure 15. Alarm Data Model

7.2.1 Alarm

7.2.1.1 Type Alarm

Description: A definition of an Alarm.

Inherits from:

- Alarm_Common

Name	Type	M/O	Description	Mplify 133.1
alarmedObject	AlarmedObjectRef[]	M	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

Name	Type	M/O	Description	Mplify 133.1
isRootCause	boolean	M	Indicates whether the alarm is a root cause alarm.	Is Root Cause
parentAlarm	AlarmRef	O		Parent Alarm
probableCause	ProbableCause	O		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 AlarmedObjectRef

Description:

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

7.2.1.3 Type AlarmRef

Description:

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

7.2.1.4 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	Mplify 133.1
@type	string	M	The named type must be a subclass of AlarmSpecificAttributes.	

7.2.1.5 `enum` AlarmState

Description: Defines the alarm state during its life cycle.

Value	Mplify 133.1
acknowledged	Acknowledged
cleared	Cleared
unAcknowledged	Unacknowledged

7.2.1.6 `enum` AlarmType

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

state	Mplify 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.

state	Mplify 133 name	Description
`mechanismViolation`	Mechanism Violation	A security attack has been detected by a security mechanism.
`timeDomainViolation`	Time Domain Violation	An event has occurred at an unexpected or prohibited time.

7.2.1.7 Type Alarm_Common

Description: A definition of an Alarm.

Alarm Changed Time

Name	Type	M/O	Description	Mplify 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.	
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
alarmRaisedTime	date-time *format = date-time*	M	The time that an alarm was raised. This time may differ from the Alarm Reported Time	Alarm Raised Time

Name	Type	M/O	Description	Mplify 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 alarmRaisedTime. 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
planned OutageIndicator	Planned OutageIndicator	O		Planned Outage Indicator
reportingSystemId	string	O	Reporting system identity.	Reporting System Identifier
serviceAffecting	boolean	M	Indicates whether the alarm affects service or not.	Service Affecting
state	AlarmState	M		State

7.2.1.8 Type Comment

Description: A comment entered on the alarm

Name	Type	M/O	Description	Mplify 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	User Identifier

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	Mplify 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

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
inPlannedMaintenance
outOfService

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
adapterError
applicationSubsystemFailure
bandwidthReduced
callEstablishmentError
communicationsProtocolError
communicationsSubsystemFailure
configurationOrCustomizationError
congestion

Value
corruptData
cpuCyclesLimitExceeded
datasetOrModemError
degradedSignal
dteDceInterfaceError
enclosureDoorOpen
equipmentMalfunction
excessiveVibration
fileError
fireDetected
floodDetected
framingError
heatingVentilationCoolingSystemProblem
humidityUnacceptable
ioDeviceError
inputDeviceError
lanError
leakDetected
localNodeTransmissionError
lossOfFrame
lossOfSignal
materialSupplyExhausted
multiplexerProblem
outOfMemory
outputDeviceError
performanceDegraded
powerProblem
pressureUnacceptable
processorProblem
pumpFailure
queueSizeExceeded
receiveFailure
receiverFailure
remoteNodeTransmissionError
resourceNearingCapacity
responseTimeExcessive
retransmissionRateExcessive
softwareError

Value
softwareProgramTerminated
softwareProgramError
storageCapacityProblem
temperatureUnacceptable
thresholdCrossed
timingProblem
toxicLeakDetected
transmitFailure
transmitterFailure
underlyingResourceUnavailable
versionMismatch

7.2.1.12 Type ServiceRef

Description: Reference to a Service instance.

Name	Type	M/O	Description	Mplify 133.1
href	string	O	Hyperlink reference to Service	
id	string	M	unique identifier of Service	

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	Mplify 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

Name	Type	M/O	Description	Mplify 133.1
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	Mplify 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	Mplify 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
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
numberOfPmMetric CalculationIntervals	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.

7.2.3.2. Type EventSubscription

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

Inherits from:

- EventSubscriptionInput

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

7.2.3.1. Type EventSubscriptionInput

Description: This class is used to register for Notifications.

Name	Type	M/O	Description	Mplify 133.1

Name	Type	M/O	Description	Mplify 133.1
callback	string	M	This callback value must be set to *host* property from Alarm Notification API (alarmNotification.api.yaml). This property is appended with the base path and notification resource path specified in that API to construct an URL to which notification is sent. E.g. for 'callback': "https://buyer.mplify.com/ listenerEndpoint", the alarm event notification will be sent to: `https://buyer.mplify.com/listenerEndpoint/ mefApi/legato/alarmNotification/v3/listener/`	
query	string	O	This attribute is used to define which type of events to register to. Example: 'query':'eventType = createAlarm'. To subscribe for more than one event type, put the values separated by a comma: `eventType=alarmCreateEvent, alarmAttributeValueChangeEvent`. The possible values are enumerated by Event type enums in alarmNotification.api.yaml. An empty query is treated as specifying no filters - ending in subscription for all event types.	

7.3. Notification API Data model

Figure 16 presents the Alarm Notification data model.

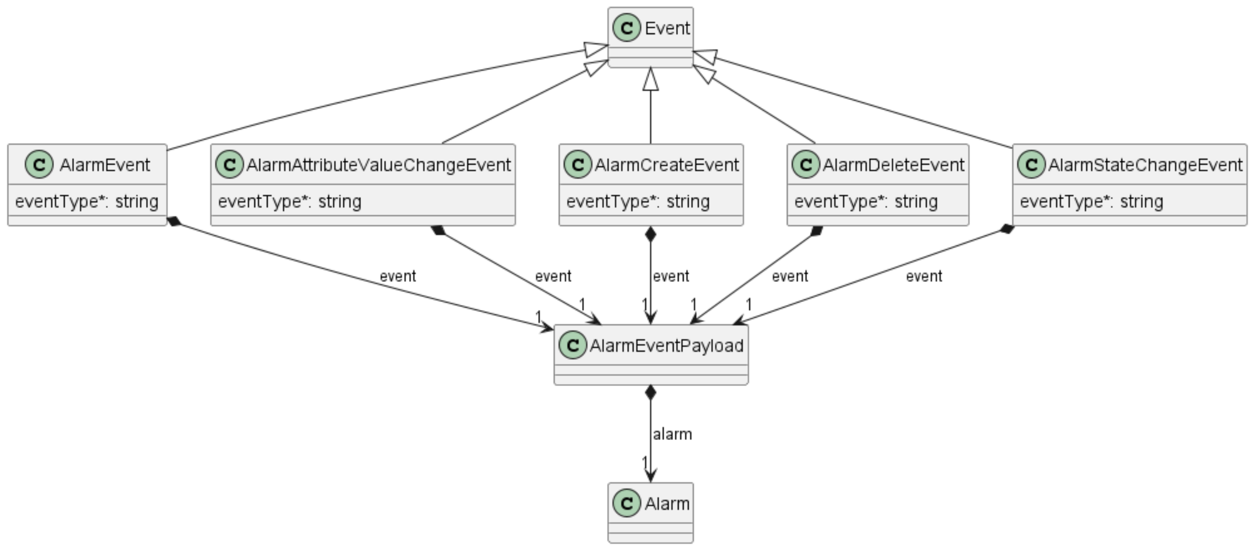


Figure 16. Alarm Notification Data Model

This data model is used to construct requests and responses of the API endpoints described in 5.2.2. Buyer/Client side API Endpoints.

7.3.1. Type Event

Description: Event class is used to describe information structure used for notification.

Name	Type	M/O	Description	Mplify 133.1

Name	Type	M/O	Description	Mplify 133.1
eventId	string	M	Id of the event	
eventTime	date-time *format = date-time*	M	Date-time when the event occurred	
eventType	string	M	The type of the notification.	
event	object	M	The event linked to the involved resource object	

7.3.2. Type AlarmAttributeValueChangeEvent

Description:

Inherits from:

- Event

Name	Type	M/O	Description	Mplify 133.1
eventType	string	M	Indicates the type of the event.	
event	AlarmEventPayload	M	A reference to the object that is source of the notification.	

7.3.3. Type AlarmCreateEvent

Description:

Inherits from:

- Event

Name	Type	M/O	Description	Mplify 133.1
eventType	string	M	Indicates the type of the event.	
event	AlarmEventPayload	M	A reference to the object that is source of the notification.	

7.3.4. Type AlarmDeleteEvent

Description:

Inherits from:

- Event

Name	Type	M/O	Description	Mplify 133.1
eventType	string	M	Indicates the type of the event.	
event	AlarmEventPayload	M	A reference to the object that is source of the notification.	

7.3.5. Type AlarmEvent

Description:

Inherits from:

- Event

Name	Type	M/O	Description	Mplify 133.1
eventType	string	M	Indicates the type of the event.	
event	AlarmEventPayload	M	A reference to the object that is source of the notification.	

7.3.6. Type AlarmEventPayload

Description: The identifier of the Test Job being subject of this event.

Name	Type	M/O	Description	Mplify 133.1
alarm	Alarm	M		

7.3.7. Type AlarmStateChangeEvent

Description:

Inherits from:

- Event

Name	Type	M/O	Description	Mplify 133.1
eventType	string	M	Indicates the type of the event.	
event	AlarmEventPayload	M	A reference to the object that is source of the notification.	

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.
- MEF 128.1, LSO API Security Profile, April 2024
- Mplify133.1 Allegro, Interlude and Legato Fault Management and Alarm API BR&UC, October 2025
- OAS-v3, February 2020
- [REST] Chapter 5: Representational State Transfer (REST) Fielding, Roy Thomas, Architectural Styles and the Design of Network-based Software Architectures (Ph.D.).
- RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, by S. Bradner, March 1997
- RFC 3986 Uniform Resource Identifier (URI): Generic Syntax, January 2005
- 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.
- TMF 630 TMF630 REST API Design Guidelines 4.2.0
- TMF 642, TMF642 Alarm User Guide v5.0.0, September 2023

Appendix A Acknowledgments

Mike **BENCHECK**

Michał **ŁĄCZYŃSKI**

Dominik **OGRODNIK**

Jack **PUGACZEWSKI**

Boris **TRINAJSTIC**