MEF Standard

MEF 135

LSO Legato Service Inventory Management API - Developer Guide

October 2023

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 members 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.

Table of Contents

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
Amartus
Cisco
Lumen
Verizon

Table 1. Contributing Members

1. Abstract

This standard is intended to assist the implementation of the Application Programming Interfaces (APIs) for the Service Inventory function of the Service Orchestration Functionality at the LSO Legato Interface Reference Point. The Legato Interface Reference Point is defined in the MEF 55.1 [MEF55.1] at the interface between the Business Application Systems layer and Service Orchestration Functionality layer.

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

MEF-LSO-Legato-SDK

commit id: 0e83943f529e87c036a083926a1b28a0a3523c5e

- `serviceApi/inventory/serviceInventoryManagement.api.yaml`
- `serviceApi/inventory/serviceInventoryNotification.api.yaml`

2. Terminology and Abbreviations

This section defines the terms used in this document. In many cases, the normative definitions of terms are found in other documents. In these cases, the third column is used to provide the reference that is controlling, in other MEF or external documents.

In addition, terms defined in the following documents are included in this document by reference, and are not repeated in the table below.

- MEF 55.1, *Lifecycle Service Orchestration (LSO): Reference Architecture and Framework* February 2021 [MEF 55.1]

Term	Definition	Source
API Endpoint	The endpoint of a communication channel (the complete URL of an API Resource) to which the HTTP-REST requests are addressed in order 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 a OAS *SchemaObject* with specified *API Endpoints*	restfulapi.net This document
Business Applications	The Service Provider functionality supporting Business Management Layer functionality	MEF 55.1
OAS Document	An API description document in the OpenAPI specification format.	openapis.org
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 BUS and SOF, potentially involving multiple back and forth transactions.	This document
SchemaObject	The construct that allows the definition of input and output data types. These types can represent object classes, as well as primitives and arrays. specification	spec.openapis.org

Term	Definition	Source
Service Orchestration Functionality	The set of service management layer functionality supporting an agile framework to streamline and automate the service lifecycle in a sustainable fashion for coordinated management supporting design, fulfillment, control, testing, problem management, quality management, usage measurements, security management, analytics, and policy-based management capabilities providing coordinated end-to-end management and control of Services	MEF 55.1

Table 2. Terminology

Term	Definition	Source
API	Application Programming Interface. In this document, API is used synonymously with REST API.	This document
BUS	Business Applications	MEF 55.1
IRP	Interface Reference Point	This document
OAS	OpenAPI Specification	openapis.org
SOF	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.

4. Introduction

This standard specification document describes the Application Programming Interface (API) for Service Inventory Management functionality of the LSO Legato Interface Reference Point (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 IRP highlighted.

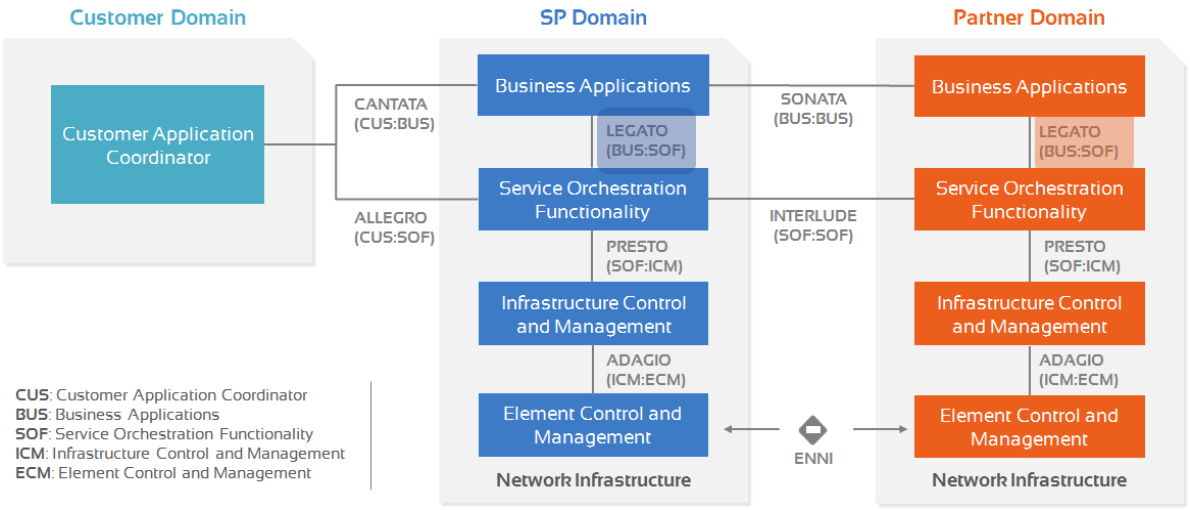

Figure 1. The LSO Reference Architecture

4.1. Description

This standard is scoped to cover APIs for following Service Orchestration Functionalities:

- Service Inventory Management
- Service Notification

Other Service Orchestration Functionalities not addressed in this standard include (but not limited to):

- Service Ordering and Fulfillment Service Catalog Management
- Service Qualification
- Service Activation Testing
- Service Problem Management
- Service Quality Management
- Service Usage measurements and Reporting (in support of billing)
- License Management

This document supports interactions over the Legato interface within a single operator. Both the Business Applications (BUS) and Service Orchestration Functionality (SOF) systems use the information contained within this document.

This standard is intended to support the design of API implementations that enable inter-operable SOF operations (in scope of this standard) across the Legato IRP.

This standard is based on TMF Open API (v4.0.0) for Service Inventory TMF 638.

The Service Inventory API allows the BUS to retrieve information about existing (previously ordered) Services from the SOF's Inventory. The SOF's Service Inventory is a set of instances of Services that have been ordered by a BUS.

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. `Service`).
- 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

The API definition builds on *TMF638 Service Inventory API REST Specification v4.0.1* [TMF 638]. Service Inventory Use Cases must support the use of any of MEF service specifications as payload, in particular, those defined in:

- *LSO Legato Service Specification - SD-WAN Schema Guide* in MEF W100 [MEF W100].
- *LSO Legato Service Specification - Carrier Ethernet Schema Guide* in MEF W101 [MEF W101].
- *LSO Legato Service Specification - IP/IP-VPN Schema Guide* in MEF W102 [MEF W102].

4.4. Approach

As presented in Figure 2. the 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)

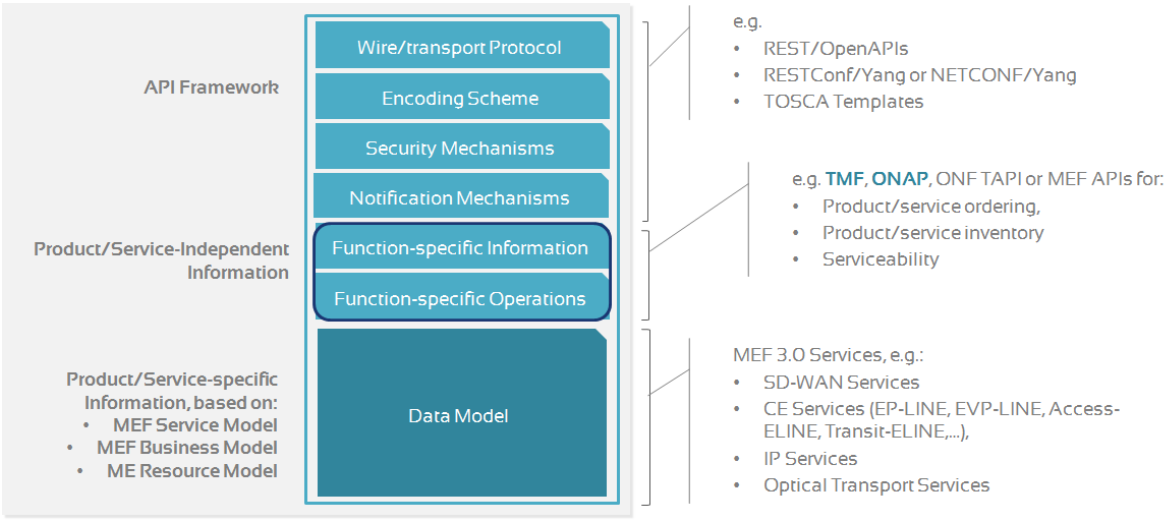

Figure 2. Legato API Structure

The essential concept behind the framework is to decouple the common structure, information, and operations from the specific service information content.
Firstly, the Generic API Framework defines a set of design rules and patterns that are applied across all Legato APIs.
Secondly, the service-independent information of the framework focuses on a model of a particular Legato functionality and is agnostic to any of the service specifications. For example, this standard is describing the Service model and operations that allow inventory queries of any service that is aligned with either MEF or custom service specifications.
Finally, the service-specific information part of the framework focuses on MEF service specifications that define business-relevant attributes and requirements for trading MEF services.

This Developer Guide is not defining MEF service specifications but can be used in combination with any service specifications defined by or compliant with MEF. Examples of MEF Service Model (MSM) schema include:

- MEF W100: SD-WAN Services based on MEF 70 [MEF70]
- MEF W101: Carrier Ethernet services based on MEF 10.4 [MEF10.4] and MEF 26.2 [MEF26.2]
- MEF W102: IP Services based on MEF 61.1 [MEF61.1] and MEF 61.1.1 [MEF61.1.1]

4.5. High-Level Flow

The Legato Service Catalog, Service Order, Service Inventory, and Service Notification APIs in essence allow the BUS to request SOF to configure and activate one or more services as part of an order fulfillment process. Figure 3 presents a high-level flow of use of all of the above-mentioned APIs.

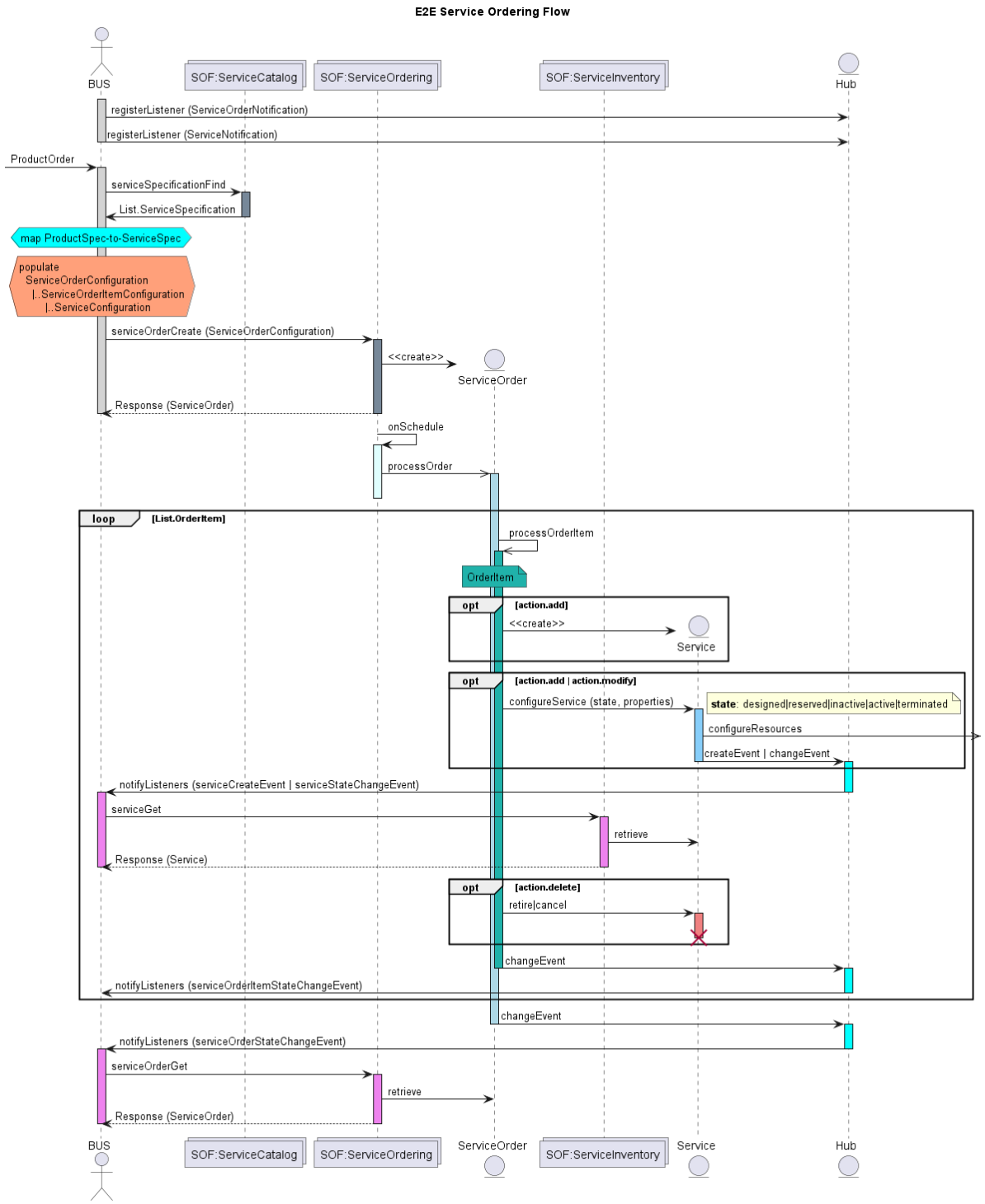

Figure 3. High-Level Flow

The following steps describe the high-level flow:

- The BUS system registers for notifications.
- As part of the ordering flow, the BUS system receives the product order (through Cantata or Sonata) which triggers the fulfillment processes in the BUS system.
- The BUS system first queries the *Service Catalog* to retrieve the `ServiceSpecifications` supported by the SOF
  Note1: *Service Catalog and the process of mapping and decomposing a product order to identify appropriate* `ServiceSpecifications` *is out of scope for this standard.* ***Note2***: *The*

mechanisms to design, construct and populate the `ServiceSpecifications` *into SOF Service Catalog is out of scope for this standard.*

  - Each specific instance of a `ServiceSpecification` (retrieved from the *Service Catalog*) minimally contains a reference to target `Service` schema. A Service schema describes the set of properties that characterize that service and are exchanged over Legato IRP.

- During the service configuration and activation phase, the BUS system uses the *Service Order API* to instantiate the `Service` utilizing the `ServiceSpecifications` (retrieved from the *Service Catalog*).
  - The BUS achieves this by creating a `ServiceOrder` which contains a one or more `ServiceOrderItems`.
  - Each `ServiceOrderItem` carries some `ServiceConfiguration` data and the type of operation (*`add`*/*`modify`*/*`delete`*) to be performed (instructions to SOF).
  - The SOF utilizes `Service` schema referenced in the `ServiceSpecification` to validate the `ServiceConfiguration` data passed in by the BUS.
  - The `ServiceOrder` / `ServiceOrderItem` is processed by the SOF as per the state transition rules described in MEF W99
  - The SOF reports the `ServiceOrder` and `ServiceOrderItem` state changes
  - The SOF performs the actions (*`add`*/*`modify`*/*`delete`*) specified in a `ServiceOrderItem` on the specified target `Service` instance in the *Service Inventory* as per the state transition rules described in 6.1.1. Service State Machine
  - The SOF reports the `Service` instance state changes
- The BUS system uses the same *Service Order API* to create **new** `Service` instances as well as update **existing** `Service` instance's properties or trigger state transitions, and delete **existing** `Service` instance.

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 service-agnostic and service-specific parts of API payloads. Finally, payload validation and API security aspects are discussed.

5.1. High-level Use Cases

Figure 4. presents a high-level use case diagram. It aims to help understand the endpoint mapping. Use cases are described extensively in chapter 6

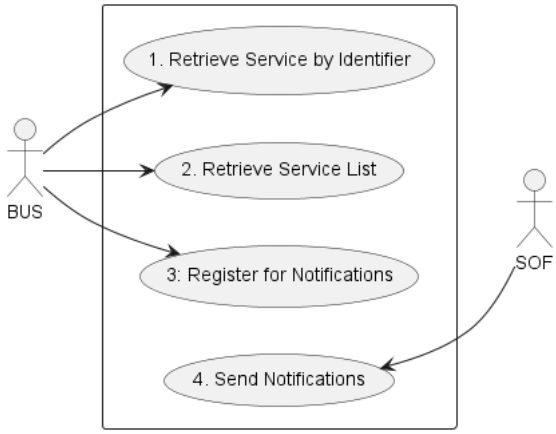

Figure 4. Use cases

5.2. API Endpoints and Operations Summary

5.2.1. SOF Service Inventory API Endpoints

Base URL: `https://{{serverBase}}:{{port}}{{?/sof_prefix}}/mefApi/legato/serviceInventory/v5/`

The following API Endpoints are used by BUS to create and query for `Service` instances and to subscribe/unsubscribe to service notifications. The endpoints and corresponding data model are defined in:

`serviceApi/inventory/serviceInventoryManagement.api.yaml`

API Endpoint	Description	Use Case mapping

API Endpoint	Description	Use Case mapping
`GET /service/{{id}}`	A request initiated by the BUS to retrieve a specific `Service` from the service inventory system in SOF, that match the *`id`* provided as *`path`* parameter	UC 1: Retrieve Service by Service Identifier
`GET /service`	A request initiated by the BUS to retrieve a list of `Services` from the service inventory system in SOF, that match the filter criteria provided as *`query`* parameters	UC 2: Retrieve List of Services
`POST /hub`	A request initiated by the BUS to instruct the SOF to send notification	UC 3: Register for Notifications
`GET /hub/{{id}}`	A request initiated by the BUS to retrieve a specific `EventSubscription` from the service order management system in SOF, that matches the provided *`id`* provided as *`path`* parameter	UC 4: Register for Notifications
`DELETE /hub/{{id}}`	A request initiated by the BUS to instruct the SOF to stop sending notifications	UC 4: Register for Notifications

Table 4. SOF Service Inventory API Endpoints

[R1] SOF **MUST** support all API endpoints listed in Table 4.

5.2.2. BUS Service Inventory API Endpoints

Base URL: `https://{{serverBase}}:{{port}}{{?/bus_prefix}}/mefApi/legato/serviceInventoryNotification/v5/`

The following API Endpoints are used by SOF to post notifications to registered BUS listeners. The endpoints and corresponding data model are defined in

`serviceApi/inventory/serviceInventoryNotification.api.yaml`

API Endpoint	Description	Use Case mapping
`POST /listener/serviceCreateEvent`	A request initiated by the SOF to notify BUS on *`Service`* instance creation	5. Send Notifications
`POST /listener/serviceDeleteEvent`	A request initiated by the SOF to notify BUS t on *`Service`* instance deletion	5. Send Notifications

API Endpoint	Description	Use Case mapping
`POST /listener/serviceStateChangeEvent`	A request initiated by the SOF to notify BUS on *ServiceO* instance state change	5. Send Notifications

Table 5. BUS Service Inventory API Endpoints

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

[O2] The BUS **MAY** register to receive service notifications.

[R2] The SOF **MUST** support sending notification to API endpoints listed in Table 5 to registered BUS.

5.3. Integration of Service Specifications into Service Inventory API

Service specifications are defined using JsonSchema (draft 7) format JSON Schema draft 7 and are integrated into the `Service` using the TMF extension pattern.

The extension hosting type in the API data model is `MefServiceConfiguration`. The `@type` attribute of that type must be set to a value that uniquely identifies the service specification. A unique identifier for MEF standard service specifications is in URN format and is assigned by MEF. This identifier is provided as root schema `$id` and in service specification documentation. Use of non-MEF standard service definitions is allowed. In such a case the schema identifier must be agreed upon between the BUS and the SOF.

The example below shows a header of a Service Specification schema, which is describing the IP Uni, where `"$id": urn:mef:lso:spec:legato:ip-uni:v0.0.1:all` is the above-mentioned URN:

```
"$schema": http://json-schema.org/draft-07/schema#
"$id": $id": urn:mef:lso:spec:legato:ip-uni:v0.0.1:all
title: MEF LSO Legato - IP UNI Specification
```

Service specifications are provided as Json schemas without the `MefServiceConfiguration` context.

Service-specific attributes are introduced to the `Service` with the `serviceConfiguration` attribute of type `MefServiceConfiguration` which is used as an extension point for service-specific attributes.

Implementations might choose to integrate selected service specifications to data model during development. In such a case an integrated data model is built and service specifications are in an inheritance relationship with `MefServiceConfiguration` as described in the OAS specification. This pattern is called **Static Binding**. The SDK is additionally shipped with a set of API definitions that statically bind all service-related APIs (POQ, Quote,

Order, Inventory) with all corresponding service specifications available in the release. The snippets below present an example of a static binding of the envelope API with several MEF service specifications, from both `MefServiceConfiguration` and service specification point of view:

```
MefServiceConfiguration:
  description:
    MefServiceConfiguration is used as an extension point for MEF-specific
    service payload. The `@type` attribute is used as a discriminator
  discriminator:
    mapping:
      urn:mef:lso:spec:legato:ip-enni:v0.0.1:all: "#/components/schemas/IpEnni"
      urn:mef:lso:spec:legato:ipvc-endpoint:v0.0.1:all: "#/components/schemas/IpvcEndpoint"
      urn:mef:lso:spec:legato:ip-uni:v0.0.1:all: "#/components/schemas/IpUni"
      urn:mef:lso:spec:legato:ethernet-uni-access-link-trunk:0.0.1:all:
"#/components/schemas/EthernetUniAccessLinkTrunk"
      urn:mef:lso:spec:legato:ip-uni-access-link:0.0.1:all: "#/components/schemas/IpUniAccessLink"
      urn:mef:lso:spec:legato:ipvc:v0.0.1:all: "#/components/schemas/Ipvc"
      urn:mef:lso:spec:legato:ip-uni-access-link-trunk.0.1:all: "#/components/schemas/IpUniAccessLinkTrunk"
      urn:mef:lso:spec:legato:ip-enni-link:v0.0.1:all: "#/components/schemas/IpEnniLink"
    propertyName: "@type"
  properties:
    "@type":
      description:
        The name of the type, defined in the JSON schema specified above, for
        the service that is the subject of the Request. The named type must be a
        subclass of MefServiceConfiguration.
      type: string
```

```
IpvcEndpoint:
  allOf:
    - $ref: "#/components/schemas/MefServiceConfiguration"
    - description:
        "An IPVC End Point is a logical entity at an EI, to which a subset of
        packets that traverse the EI is mapped. Reference MEF 61.1 Section 7.4
        IP Virtual Connections and IPVC End Points."
```

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

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

[R3] `MefServiceConfiguration` type is an extension point that **MUST** be used to integrate service specifications' properties into a request/response payload.

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

[R5] Service attributes specified in the payload must conform to the service specification specified in the `@type` property.

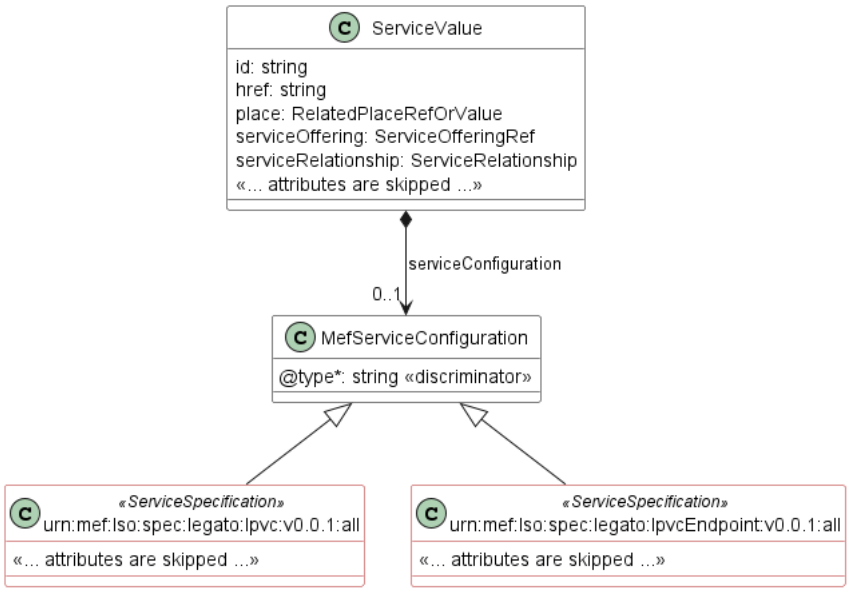

Figure 5. The Extension Pattern with Sample Service-Specific Extensions

Figure 5 presents two MEF `<<ServiceSpecifications>>` that represent IPVC and IPVC Endpoint services. When these services are used as a Service payload the `@type` of `MefServiceConfiguration` takes `"urn:mef:lso:spec:legato:ipvc:v0.0.1:all"` or `"urn:mef:lso:spec:legato:ipvc-endpoint:v0.0.1:all"` value to indicate which service specification should be used to interpret a set of service-specific attributes included in the payload. An example of service configuration is presented in Section 6.2.

The *all* suffix after the service type name in the URN comes from the approach that the service schemas may differ depending on the function (POQ, Quote, Order, or Inventory) they are used with. The value *all* means that one version of schema is shared by all functions.

5.4. Sample Service Specification

The Legato SDK contains service specification definitions, from which IPVC and IPVC End Point are used in the payload samples in this section. The schemas are located in the SDK package at:

- `serviceSchema/ip/ipvc.yaml`
- `serviceSchema/ip/ipvcEndPoint.yaml`

The service specification data model definitions are available as JsonSchema (version `draft 7`) documents. Figures 6 and 7 depict simplified UML views on these data models in which:

- the mandatory attributes are marked with `*`,
- the mandatory relations have a cardinality of `1` or `1..*`,
- some relations and attributes that are not essential to the understanding of the service specification model are omitted.

The red color in Figures 6 and 7 below highlights the data model of services. Some parts of the model are skipped for examples clarity. This is denoted by the `<<skipped>>` text in diagrams and in json snippets later in the document. Please note that this document uses service specifications just for the sake of example on how to use the Service Inventory API together with the Service payload. The detailed examples of any service specification is not in the scope of this document.

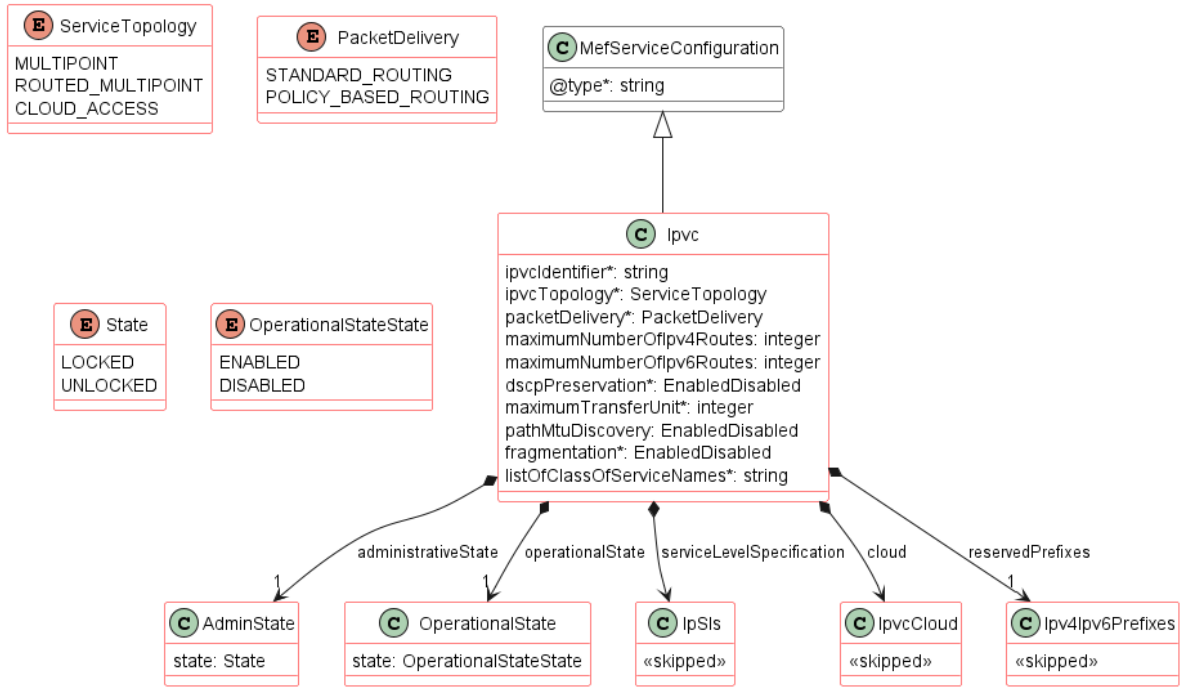

Figure 6. A simplified view on IPVC service specification data model

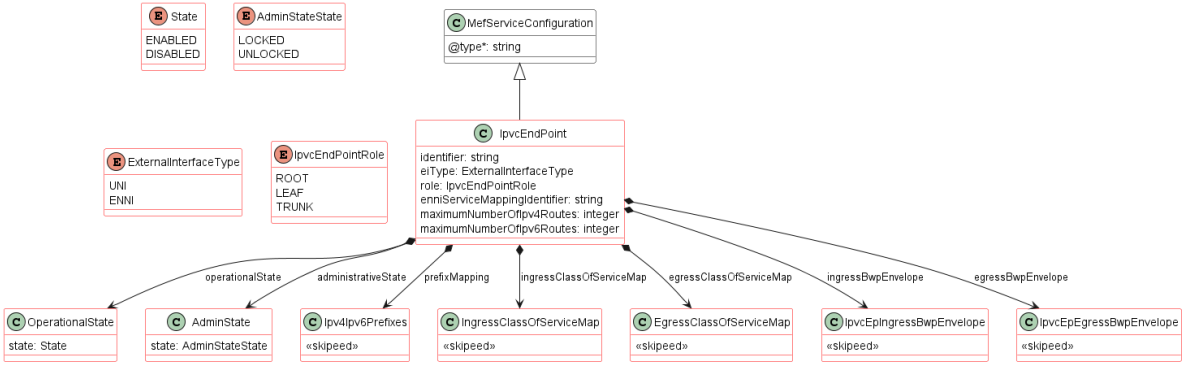

Figure 7. A simplified view of IPVC End Point service specification data model

Service specifications define several service-related and envelope-related requirements. For example:

- for an IPVC End Point service two mandatory relationships must be specified, one toward the IPVC (`IPUNI_ENDPOINT_OF_IPVC`), and a second toward the IP UNI (`CONNECTS_TO_IPUNI`)
- for an IP UNI Access Link Trunk service a place relationship (`INSTALL_LOCATION`) must be specified

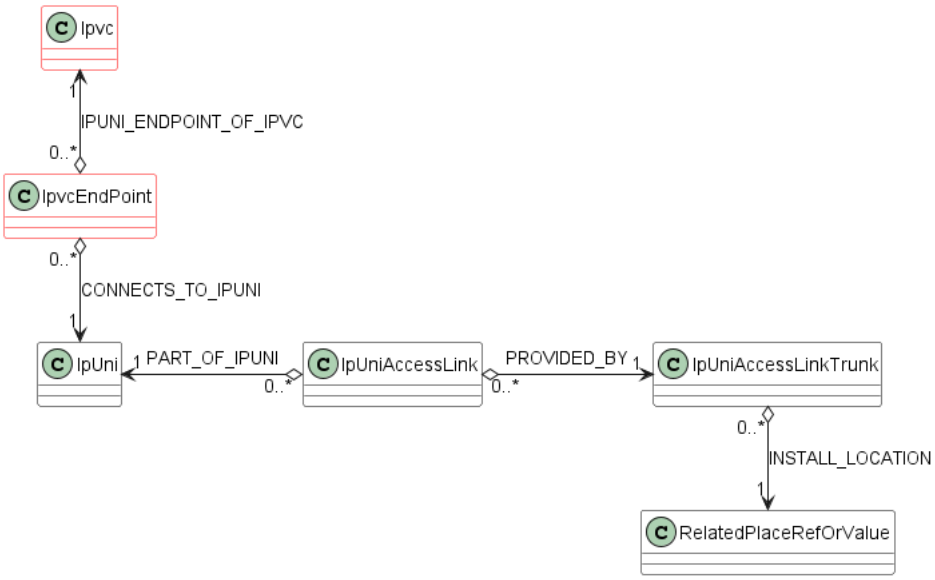

Figure 8. Example use case configuration

Figure 8 shows an example (limited to class names and relations) of a typical setup of the Advanced Internet Access service. It is built from 5 services:

- IPVC
- IPVC End Point
- IP UNI
- IP UNI Access Link
- IP UNI Access Link Trunk

The example highlights IPVC and IPVC End Point (with red lines) that were mentioned earlier. Note the relations outgoing from the `IpvcEndPoint`. The relations are provided with the use of `serviceRelationship` attribute.

5.5. Model structure and validation

The structure of the payloads exchanged via Legato Service API endpoints is defined using:

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

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

[R7] A service specification may define additional consistency rules and requirements that **MUST** be respected by implementations. These are defined for:

- required relation type, multiplicity to other services
- related contact information roles that are to be defined for a service
- relations to places (locations) and their roles that are to be defined for a service

5.6. 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 BUS or SOF is allowed to do and what information may be obtained. However, 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 *LSO API Security Profiles* [MEF128].

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 examples for each of them.

Use Case #	Use Case Name	Use Case Description
1	Retrieve Service by Service Identifier	A request initiated by the BUS to retrieve the details of a specific Service with a given Service Identifier.
2	Retrieve Service List	A request initiated by the BUS to retrieve a list of Services that match the provided filter criteria
3	Register for Notifications	The BUS requests to subscribe to notifications.
4	Send Notifications	A notification initiated by the SOF to the BUS

Table 6. Use cases description

6.1. Use case 1: Retrieve Service by Identifier

To get detailed and up-to-date information about the Service, the BUS sends a Retrieve Service by Identifier request using a `GET /service/{id}` operation.

The flow is a simple request-response pattern, as presented in Figure 9:

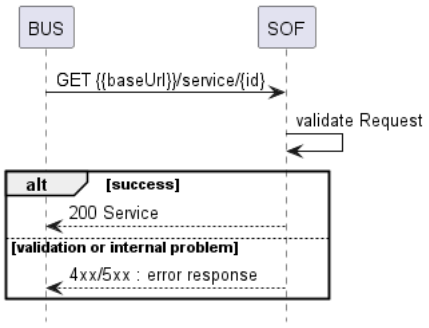

Figure 9. Use case 1: Retrieve Service by Service Identifier flow

The model taking part in this use case is presented in Figure 10:

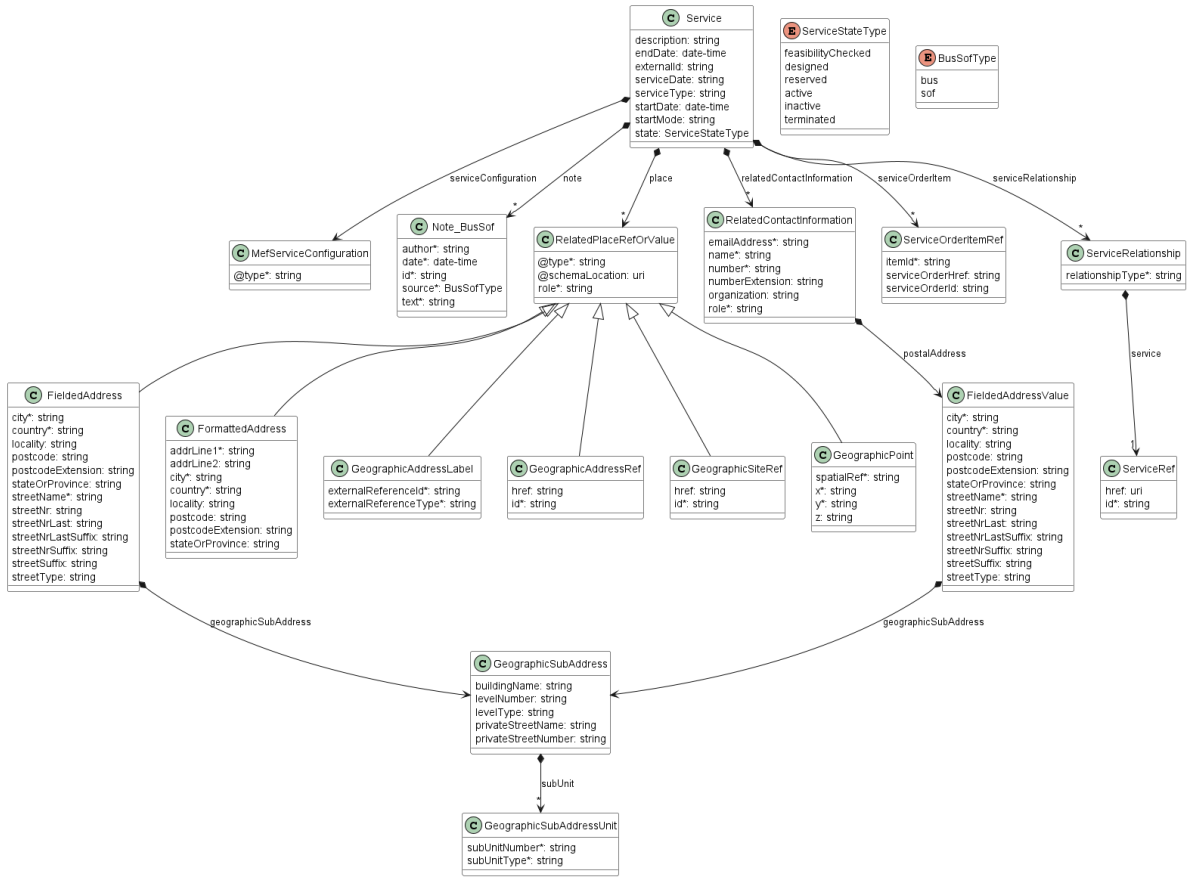

Figure 10. Use case 1: Service model

Example request and response:

```
GET /mefApi/legato/serviceInventory/v5/service/00000000-5555-6666-7777-000000009999
```

As presented in Figures 6 and 7, for readability the service examples show only first-level simple attributes. This is marked with a `<<skipped>>` label.


```
{
  "id": "00000000-5555-6666-7777-000000009999",
  "description": "IPVC End Point",
  "externalId": "BUS_IPVC_END_POINT-0001",
  "serviceType": "Internet Access",
  "state": "active",
  "name": "IPVCEndpoint",
  "serviceRelationship": [
    { << relation to IP UNI >>
      "relationshipType": "CONNECTS_TO_IPUNI",
      "service": {
        "id": "IP_UNI_0000-0001"
      }
    },
    { << relation to IPVC >>
      "relationshipType": "IPUNI_ENDPOINT_OF_IPVC",
      "service": {
        "id": "00000000-5555-6666-7777-000000008888"
      }
    }
  ],
  "serviceConfiguration": {
    "@type": "urn:mef:lso:spec:legato:ipvc-end-point:v0.0.1:all",
    "administrativeState": {
      "state": "UNLOCKED"
    },
    "operationalState": {
      "state": "ENABLED"
    },
```

```
      "identifier": "IPVC-EndPoint-0000-0001",
      "eiType": "UNI",
      "role": "ROOT",
      "prefixMapping": {}, <<skipped>>
      "maximumNumberOfIpv4Routes": 2,
      "maximumNumberOfIpv6Routes": 0,
      "ingressClassOfServiceMap": {}, <<skipped>>
      "egressClassOfServiceMap": {}, <<skipped>>
      "ingressBwpEnvelope": {}, <<skipped>>
      "egressBwpEnvelope": {} <<skipped>>
    }
  }
```

[R8] In case `id` does not allow finding a `Service` in SOF's system, an error response `Error404` **MUST** be returned.

6.1.1. Service State Machine

The Inventory reflects the actual state of the Service. The lifecycle of a Service is presented in Figure 11. The labels of the transitions are informative "use cases" names.

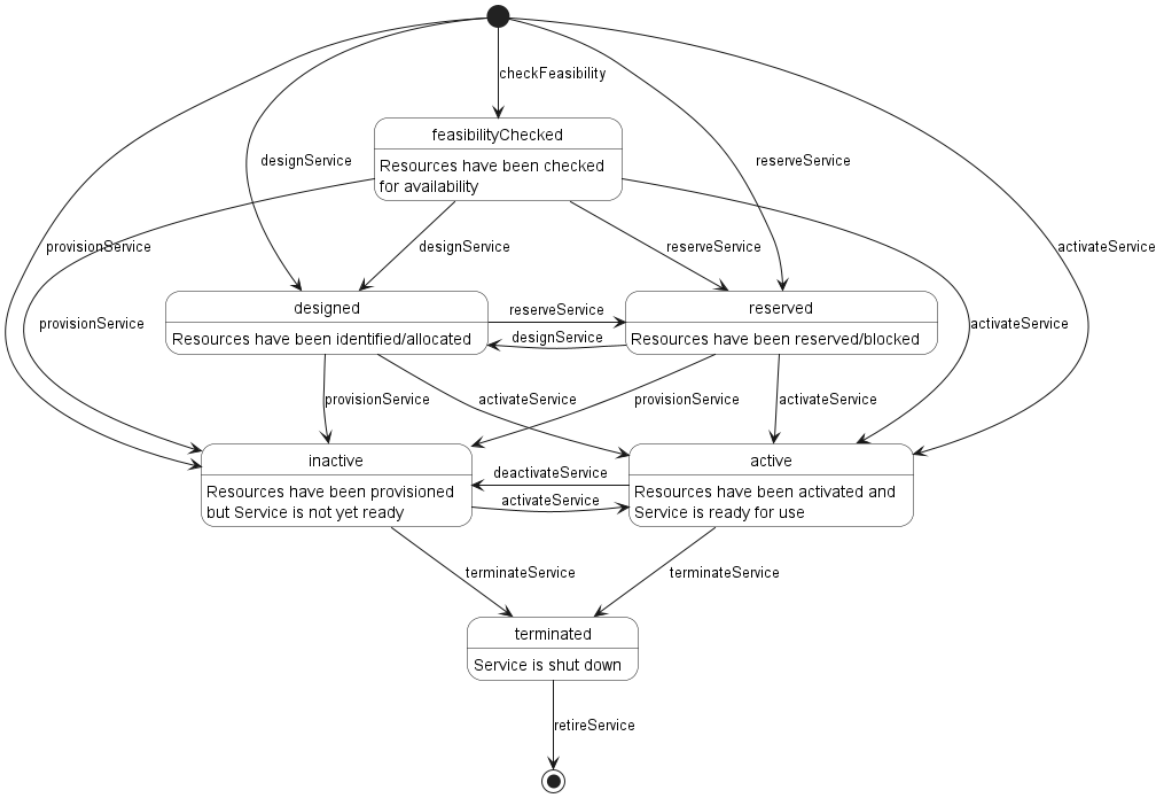

Figure 11. Service State Machine

Additions and changes to Services in the Service Inventory can be performed with use of Service Orders and the Service Order Management API, or by the request of the SOF.

A detailed description of each state can be found in the Table 7 below.

State	Description
feasibilityChecked	Initial check whether the necessary resources are available and sufficient for the installation of a given service.

State	Description
designed	The Service is designed. The resources are identified and/or allocated, but not reserved.
reserved	All required resources for the given service are reserved and ready.
inactive	The service is deactivated and is no longer available.
active	The service is fully available and active
terminated	The service is 'logically deleted'. All associated resources are freed and made available for service to other users.

Table 7. Service states

6.1.2. Specifying Place Details

Some service specifications may define requirements for place relationships. As shown in the example in Figure 8, an IP UNI Access Link Trunk service has an `INSTALL_LOCATION` place relation.

There are different formats in which place information may be provided: `MEFGeographicPoint`, `FieldedAddress`, `FormattedAddress`, `GeographicAddressLabel`, `GeographicSiteRef`, `GeographicAddressRef`. The first four of them can be used to provide place description by value. The site and address reference allow specifying the place information as a reference to previously validated address or site available through SOF's Addressing and Site API endpoints, which definition is provided in the SDK:

- `productApi/serviceability/address/geographicAddressManagement.api.yaml`
- `productApi/serviceability/site/geographicSiteManagement.api.yaml`

The Address Validation and Site APIs are standardized by:

- Address, Service Site, and Product Offering Qualification Management, Requirements and Use Cases MEF 79
- Amendment to MEF 79: Address, Service Site, and Product Offering Qualification Management, Requirements, and Use Cases MEF 79.0.1
- Amendment to MEF 79: Address Validation MEF 79.0.2
- LSO Cantata and LSO Sonata Address Management API - Developer Guide MEF 121
- LSO Cantata and LSO Sonata Site Management API - Developer Guide MEF 122

The superclass for all address types is the `RelatedPlaceRefOrValue` which adds the `role` to add more context to the specified address. To distinguish between place types the `@type` discriminator is used.

Note: The *RefOrValue* stands for a pattern where an address can be provided either by `id` (using `GeographicSiteRef` or `GeographicAddressRef`) OR by value (with use of `MEFGeographicPoint`,

`FieldedAddress`, `FormattedAddress`, `GeographicAddressLabel`). There is no way to specify an address with use both ref AND value at the same time.

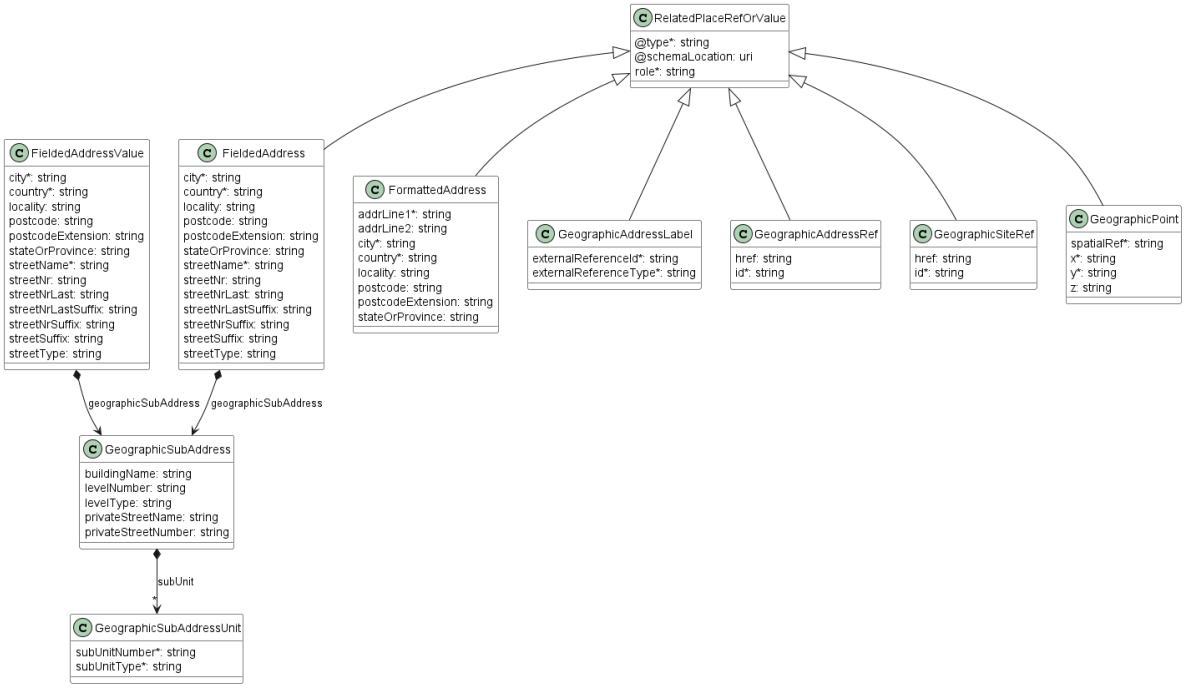

Figure 12. The data model for place representation

Examples of different place specification formats are provided below.

6.1.2.1. Fielded Address

```
{
  "@type": "FieldedAddress",
  "streetType": "ul.",
  "streetName": "Edmunda Wasilewskiego",
  "streetNr": "20",
  "streetNrSuffix": "14",
  "city": "Kraków",
  "stateOrProvince": "Lesser Poland",
  "postcode": "30-305",
  "country": "Poland",
  "geographicSubAddress": {
    "levelType": "floor",
    "levelNumber": "4"
  },
  "role": "INSTALL_LOCATION"
}
```

Fielded address example of a place specification. The type discriminator has the value `FieldedAddress`. A subset of available attributes is used to describe the place. The fielded address has an optional `geographicSubAddress` structure that defines several attributes that can be used in case precise address information has to be provided. In the example above, a floor in the building at the given address is specified using this structure. The role of the place is assigned according to the requirements of the Operator UNI service specification.

6.1.2.2. Formatted Address

```
{
  "@type": "FormattedAddress",
  "addrLine1": "ul. Edmunda Wasilewskiego 20/14",
  "addrLine2": "Floor 4",
  "city": "Kraków",
  "stateOrProvince": "Lesser Poland",
  "postcode": "30-305",
  "country": "Poland",
  "role": "INSTALL_LOCATION"
}
```

Place information in a form of a formatted address. The type discriminator has the value `FormattedAddress`. This example contains the same information as the previous `FieldedAddress` example.

6.1.2.3. Geographic Point

```
{
  "@type": "MEFGeographicPoint",
  "spatialRef": "EPSG:4326 WGS 84",
  "x": "50.048868",
  "y": "19.929523",
  "role": "INSTALL_LOCATION"
}
```

Place information in a form of a geographic point. `spatialRef` determines the standard that has to be used to interpret coordinates provided in the required `x` (latitude), `y` (longitude), and optional `z` (elevation) values.

This type allows only providing a point. It cannot carry more detailed information like the floor number from previous examples.

[R9] The `spatialRef` value that can be used **MUST** be agreed between BUS and SOF.

6.1.2.4. Geographic Address Label

```
{
  "@type": "GeographicAddressLabel",
  "externalReferenceType": "CLLI",
  "externalReferenceId": "PLTXCL01",
  "role": "INSTALL_LOCATION"
}
```

The Geographic Address Label represents a unique identifier controlled by a generally accepted independent administrative authority that specifies a fixed geographical location. The example above is a place that represents a CLLI (Common Language Location Identifier) identifier which is commonly used to refer locations in North America for network equipment installations.

6.1.2.5. Geographic Site Reference

```
{
  "@type": "GeographicSiteRef",
  "id": "18d3bb74-997a-4a62-8198-84250766765a",
  "role": "INSTALL_LOCATION"
}
```

`GeographicSiteRef` type is used to specify a `GeographicSite` by reference in the request. In the above example, a `GeographicSite` identified as `18d3bb74-997a-4a62-8198-84250766765a` in the SOFs Service Site API is used.

6.1.2.6. Geographic Address Reference

```
{
  "@type": "GeographicAddressRef",
  "id": "8198bb74-18d3-9ef0-4913-66765a842507",
  "role": "INSTALL_LOCATION"
}
```

`GeographicAddressRef` type is used to specify a `GeographicAddress` by reference in the request. In the above example, a `GeographicAddress` identified as `8198bb74-18d3-9ef0-4913-66765a842507` in the SOFs Service Site API is used.

6.2. Use case 2: Retrieve Service List

The BUS can retrieve a list of `Services` by using a `GET /service` operation with desired filtering criteria.

[O3] The BUS's request **MAY** contain none or more of the following attributes:

- `state`
- `serviceDate.lt`
- `serviceDate.gt`
- `startDate.lt`
- `startDate.gt`
- `endDate.lt`
- `endDate.gt`
- `serviceOrder.id`
- `serviceOrderItem.id`
- `externalId`
- `geographicSite.id`
- `geographicAddress.id`
- `serviceType`
- `startMode`

The flow is a simple request-response pattern, as presented in Figure 13:

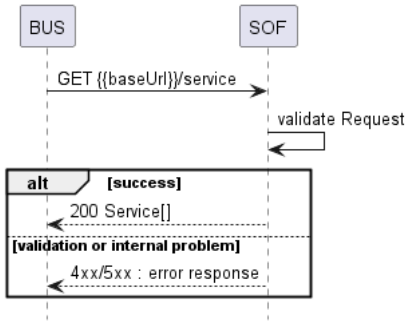

Figure 13. Use case 2: Retrieve Service List flow

The response is a list of `Service` instances, which model is the same as in the retrieve by identifier use case and is presented in Figure 10.

```
https://serverRoot/mefApi/legato/serviceInventory/v5/service?status=avtive
```

The example above shows a BUS's request to get all `Services` that are in the `active` status. The correct response (HTTP code `200`) in the response body contains a list of `Service` objects matching the criteria.

The snippet below shows an example of a response with 1 service matched:

```
[
  {
    "id": "00000000-5555-6666-7777-000000009999",
    "description": "IPVC End Point",
    "externalId": "BUS_IPVC_END_POINT-0001",
    "serviceType": "Internet Access",
    "state": "active",
    "name": "IPVCEndpoint",
    "serviceRelationship": [
      { << relation to IP UNI >>
        "relationshipType": "CONNECTS_TO_IPUNI",
        "service": {
          "id": "IP_UNI_0000-0001"
        }
      },
      { << relation to IPVC >>
        "relationshipType": "IPUNI_ENDPOINT_OF_IPVC",
        "service": {
          "id": "00000000-5555-6666-7777-000000008888"
        }
      }
    ],
    "serviceConfiguration": {
      "@type": "urn:mef:lso:spec:legato:ipvc-end-point:v0.0.1:all",
      "administrativeState": {
        "state": "UNLOCKED"
      },
      "operationalState": {
        "state": "ENABLED"
      },
      "identifier": "IPVC-EndPoint-0000-0001",
      "eiType": "UNI",
      "role": "ROOT",
      "prefixMapping": {}, <<skipped>>
      "maximumNumberOfIpv4Routes": 2,
      "maximumNumberOfIpv6Routes": 0,
      "ingressClassOfServiceMap": {}, <<skipped>>
      "egressClassOfServiceMap": {}, <<skipped>>
      "ingressBwpEnvelope": {}, <<skipped>>
      "egressBwpEnvelope": {} <<skipped>>
    }
  },
```

```
    {
      "description": "IP Virtual Connection",
      "externalId": "BUS_IPVC-0001",
      "serviceType": "Internet Access",
      "name": "IPVC",
      "state": "active",
      "relatedContactInformation": [
        {
          "emailAddress": "BUS.ServiceOrderItemContact@example.com",
          "name": "BUS Service Order Item Contact",
          "number": "+12-345-678-90",
          "role": "busServiceOrderItemContact"
        }
      ],
      "serviceConfiguration": {
        "@type": "urn:mef:lso:spec:legato:ipvc:v0.0.1:all",
        "administrativeState": {
          "state": "UNLOCKED"
        },
        "operationalState": {
          "state": "ENABLED"
        },
        "ipvcIdentifier": "IPVC-0000-0001",
        "ipvcTopology": "CLOUD_ACCESS",
        "packetDelivery": "STANDARD_ROUTING",
        "maximumNumberOfIpv4Routes": 1,
        "maximumNumberOfIpv6Routes": 0,
        "dscpPreservation": "ENABLED",
        "serviceLevelSpecification": {}, <<skipped>>
        "maximumTransferUnit": 1522,
        "pathMtuDiscovery": "ENABLED",
        "fragmentation": "DISABLED",
        "cloud": {}, <<skipped>>
        "reservedPrefixes": {}, <<skipped>>
        "listOfClassOfServiceNames": ["low"]
      }
    }
  ]
```


[R10] The BUS **MUST** be able to perform BUS Inventory Query without any filter criteria.

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

[O4] The SOF **MAY** place a limit on the length of the list returned.

[O5] If the BUS Inventory Query exceeds that length, the SOF **MAY** return an error (`Error422`) indicating that the list is too long.

A response to retrieve a list of results can be paginated. The BUS can specify following query attributes related to pagination:

- `limit` - number of expected list items
- `offset` - offset of the first element in the result list

The filtering and pagination attributes must be specified in URI query format RFC3986. The SOF returns a list of elements that comply with the requested `limit`. If the requested `limit` is higher than the supported list size the smaller list result is returned. In that case, the size of the result is returned in the header attribute `X-Result-Count`. The SOF can indicate that there are additional results available using:

- `X-Total-Count` header attribute with the total number of available results

- `X-Pagination-Throttled` header set to `true`

```
https://serverRoot/mefApi/legato/serviceInventory/v5/service?state=active&limit=10&offset=0
```

The example above shows a BUS's request to get all `Services` that are in the `active` state. Additionally, the BUS 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 `Service` objects matching the criteria.

6.3. Use case 3: Register for Notifications

The SOF communicates with the BUS with Notifications provided that:

- BUS supports a notification mechanism
- BUS has registered to receive notifications from the SOF

[O6] BUS **MAY** register for Notifications.

Supporting Notification is mandatory for SOF.

To register for notifications the BUS uses the `registerListener` operation from the API: `POST /hub`. The request contains only 2 attributes:

- `callback` - mandatory, to provide the callback address the events will be notified to,
- `query` - optional, to provide the required types of event.

Figure 14 shows all entities involved in the Notification use cases.

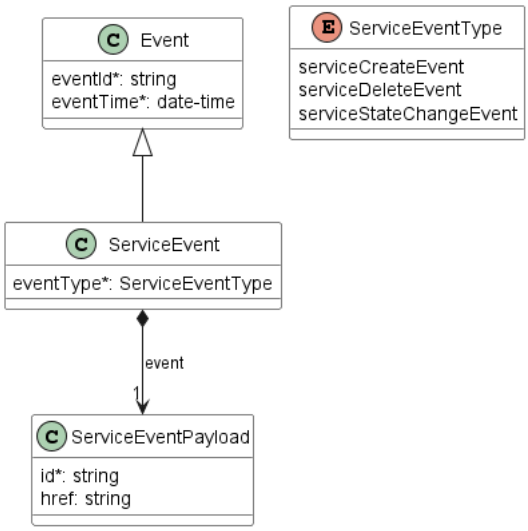

Figure 14. Service Inventory Notification Data Model

By using a simple request:

```
{
  "callback": "https://bus.com/listenerEndpoint"
}
```

The BUS subscribes for notification of all types of events. Those are:

- `serviceCreateEvent`
- `serviceDeleteEvent`
- `serviceStateChangeEvent`

If the BUS wishes to receive only notifications of a certain type, a `query` must be added:

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

If the BUS wishes to subscribe to 2 different types of events, there are 2 possible syntax variants [TMF630]:

```
eventType=serviceStateChangeEvent,serviceDeleteEvent
```

or

```
eventType=serviceStateChangeEvent&eventType=serviceDeleteEvent
```

The `query` formatting complies with RFC3986 RFC3986. According to it, every attribute defined in the Event model (from notification API) can be used in the `query`. However, this standard requires only `eventType` attribute to be supported.

[R12] `eventType` is the only attribute that the SOF **MUST** support in the query.

The SOF 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",
  "callback": "https://bus.com/listenerEndpoint",
  "query": "eventType=serviceStateChangeEvent"
}
```

Example of a final address that the Notifications will be sent to (for `serviceStateChangeEvent`):

- `https://bus.com/listenerEndpoint/mefApi/legato/serviceInventoryNotification/v5/listener/serviceStateChangeEvent`

6.4. Use case 4: Send Notification

Notifications are used to asynchronously inform the BUS about the respective objects and attributes changes.

Note: The state change notification is sent only when the state attribute actually changes its value. There are no status change notifications sent upon Service creation.

[R13] The SOF **MUST NOT** send Notifications to BUS that have not registered for them.

[R14] The SOF **MUST** send Notifications to BUS that have registered for them.

Following snippets present example of `serviceStateChangeEvent` a

```
{
  "eventId": "event-001",
  "eventType": "serviceStateChangeEvent",
  "eventTime": "2022-12-28T20:45:24.796Z",
  "event": {
    "id": "00000000-5555-6666-7777-000000009999"
  }
}
```

Note: the body of the event carries only the source object's `id`. The BUS needs to query it later by `id` to get details.

To stop receiving events, the BUS has to use the `unregisterListener` operation from the `DELETE /hub/{id}` endpoint. The `id` is the identifier received from the SOF during the listener registration.

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 Service Inventory API uses the error responses as 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 a case, the error message body structure might be aligned with the `Error`.

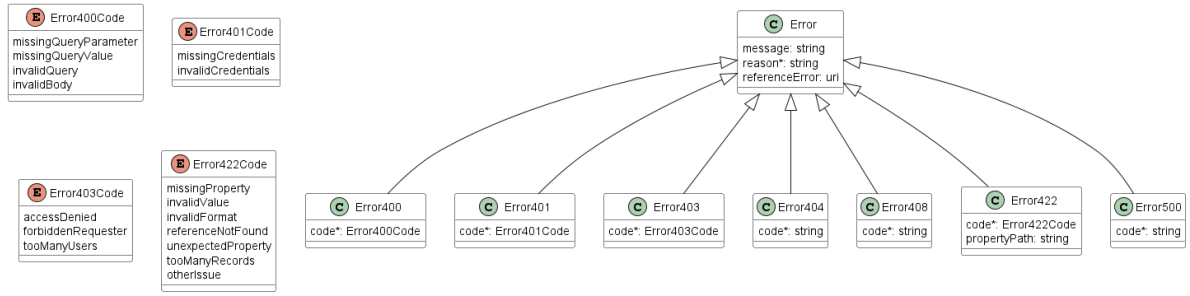

Figure 15. Data model types to represent an erroneous response

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	Text that explains the reason for the error. This can be shown to a client user.
referenceError	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	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

Value
missingQueryParameter
missingQueryValue
invalidQuery
invalidBody

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

Value
missingCredentials
invalidCredentials

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

Value
accessDenied
forbiddenRequester
tooManyUsers

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 BUS 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	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 - tooManyRecords: the number of records to be provided in the response exceeds the 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).

7.1.1.10. `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
- tooManyRecords: the number of records to be provided in the response exceeds the threshold.
- otherIssue: Other problem was identified (detailed information provided in a reason)

Value
missingProperty
invalidValue
invalidFormat
referenceNotFound
unexpectedProperty
tooManyRecords
otherIssue

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.

7.2. Management API Data model

Figure 16 presents the whole Service Inventory data model. The data types are discussed later in this section.

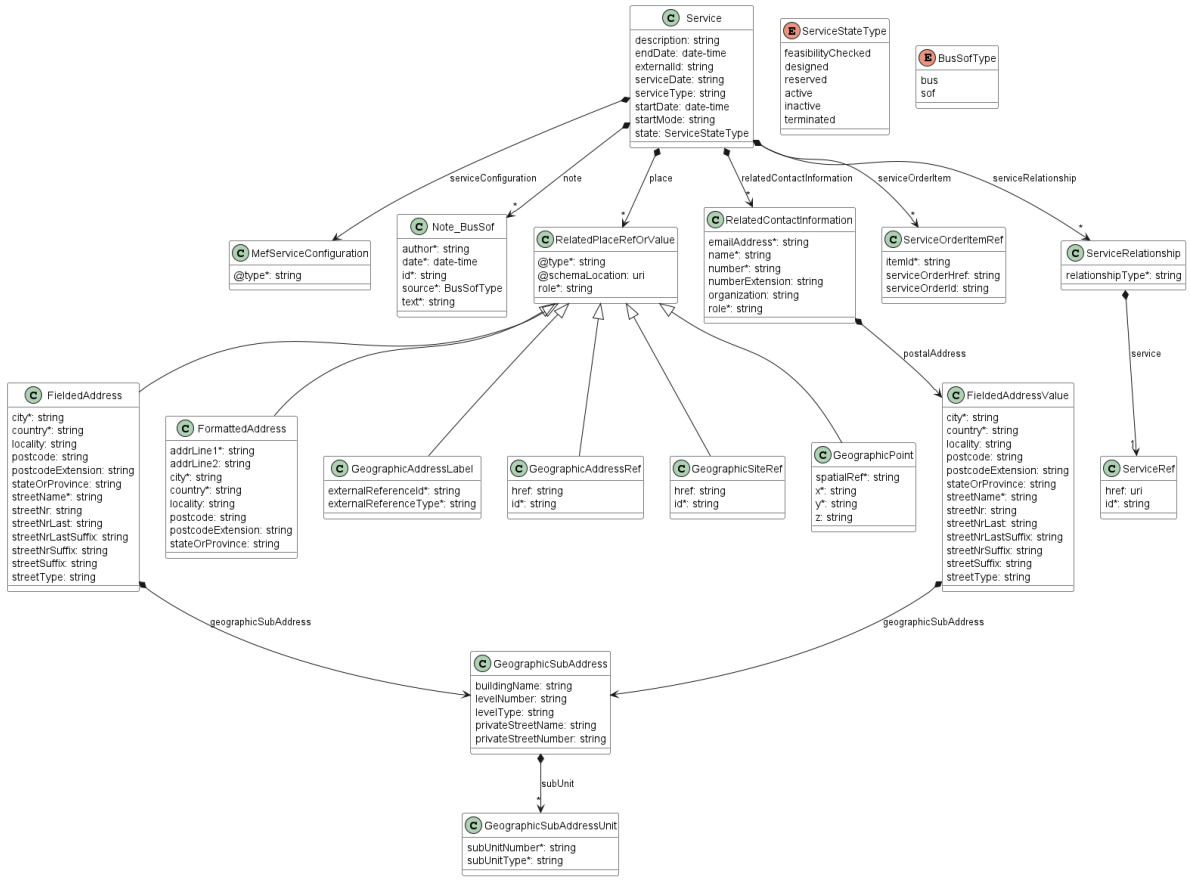

Figure 16. Service Inventory Data Model

7.2.1. Service

7.2.1.1 Type Service

Description: The Service instance managed by SOF and retrievable by an BA over the Legato IRP via the Service Inventory API.

Name	Type	Multiplicity	Description
description	string	0..1	Free-text description of the service
endDate	date-time	0..1	Date when the service ends
externalId	string	0..1	ID given by the consumer to facilitate searches
note	Note_BusSof[]	0..*	Extra-information about the order; e.g. useful to add extra delivery information that could be useful for a human process

Name	Type	Multiplicity	Description
place	RelatedPlaceRefOrValue[]	0..*	The relationships between this Service Order Item and one or more Places as defined in the Service Specification.
relatedContactInformation	RelatedContactInformation[]	0..*	Contact information of an individual or organization playing a role for this Service.
serviceConfiguration	MefServiceConfiguration	0..1	MefServiceConfiguration is used to specify the MEF specific service payload. This field MUST be populated for all item 'actions' other than 'delete'. It MUST NOT be populated when an item `action` is `delete`. The @type is used as a discriminator.
serviceDate	string	0..1	Date when the service was created (whatever its status).
serviceOrderItem	ServiceOrderItemRef[]	0..*	A list of service order items related to this service
serviceRelationship	ServiceRelationship[]	0..*	A list of service relationships. Describes links with other service(s) in the inventory.
serviceType	string	0..1	Business type of the service
startDate	date-time	0..1	Date when the service starts

Name	Type	Multiplicity	Description
startMode	string	0..1	This attribute is an enumerated integer that indicates how the Service is started, such as: 0: Unknown; 1: Automatically by the managed environment; 2: Automatically by the owning device; 3: Manually by the Provider of the Service; 4: Manually by a Customer of the Provider; 5: Any of the above
state	ServiceStateType	0..1	The life cycle state of the service.

7.2.1.2. enum ServiceStateType

Description: List of possible state for the Service.

Value	Description
feasibilityChecked	Initial check whether the necessary resources are available and sufficient for the installation of a given service.
designed	The Service is designed. The resources are identified and/or allocated, but not reserved.
reserved	All required resources for given service are reserved and ready.
inactive	The service is deactivated and is no longer available.
active	The service is fully available and active
terminated	The service is 'logically deleted'. All associated resources are freed and made available for service to other users.

7.2.1.3. Type ServiceRelationship

Description: A relationship to an existing Service. The requirements for usage for given Service are described in the Service Specification.

Name	Type	Multiplicity	Description

Name	Type	Multiplicity	Description
relationshipType*	string	1	Specifies the type (nature) of the relationship to the related Service. The nature of required relationships varies for Services of different types. For example, a UNI or ENNI Service may not have any relationships, but an Access E-Line may have two mandatory relationships (related to the UNI on one end and the ENNI on the other). More complex Services such as multipoint IP or Firewall Services may have more complex relationships. As a result, the allowed and mandatory `relationshipType` values are defined in the Service Specification.
service*	ServiceRef	1	A reference to a Service

7.2.1.4. Type ServiceOrderItemRef

Description: A reference to a Service Order Item. When referencing item from within the same Service Order, the `serviceOrderId` and `serviceOrderHref` MUST be empty.

Name	Type	Multiplicity	Description
itemId*	string	1	Identifier of referenced item within the referenced Service Order
serviceOrderHref	string	0..1	Link to the order to which the referenced item belongs to
serviceOrderId	string	0..1	Identifier of the order to which the referenced item belongs to

7.2.1.5. Type ServiceRef

Description: Reference to a Service instance.

Name	Type	Multiplicity	Description
href	uri	0..1	Hyperlink reference to Service
id*	string	1	unique identifier of Service

7.2.1.6. Type MefServiceConfiguration

Description: MefServiceConfiguration is used as an extension point for MEF specific service payload. The `@type` attribute is used as a discriminator

Name	Type	Multiplicity	Description
@type*	string	1	The value of the "$id" as defined in the JSON schema of the service.

7.2.2. Place representation

There are several formats in which place information can be provided for a Service. They are described in Section 6.1.2.

7.2.2.1. Type RelatedPlaceRefOrValue

Description: A Place provided either by value or by reference

Name	Type	Multiplicity	Description
@type*	string	1	This field is used as a discriminator and is used between different place representations. This type might discriminate for additional related place as defined in '@schemaLocation'.
@schemaLocation	uri	0..1	A URI to a JSON-Schema file that defines additional attributes and relationships. May be used to define additional related place types.
role*	string	1	Role of this place

7.2.2.2. Type FieldedAddress

Description: A type of Address that has a discrete field and value for each type of boundary or identifier down to the lowest level of detail. For example "street number" is one field, "street name" is another field, etc. Reference: MEF 79 (Sn 8.9.2)

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Multiplicity	Description
city*	string	1	The city that the address is in
country*	string	1	Country that the address is in

Name	Type	Multiplicity	Description
geographicSubAddress	GeographicSubAddress	0..1	Additional fields used to specify an address, as detailed as possible.
locality	string	0..1	The locality that the address is in
postcode	string	0..1	Descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as zip code)
postcodeExtension	string	0..1	An extension of a postal code. E.g. the part following the dash in a US urban property address
stateOrProvince	string	0..1	The State or Province that the address is in
streetName*	string	1	Name of the street or other street type
streetNr	string	0..1	Number identifying a specific property on a public street. It may be combined with streetNrLast for ranged addresses. MEF 79 defines it as required however as in certain countries it is not used we make it optional in API.
streetNrLast	string	0..1	Last number in a range of street numbers allocated to a property
streetNrLastSuffix	string	0..1	Last street number suffix for a ranged address
streetNrSuffix	string	0..1	The first street number suffix

Name	Type	Multiplicity	Description
streetSuffix	string	0..1	A modifier denoting a relative direction
streetType	string	0..1	The type of street (e.g., alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf)

7.2.2.3. Type FieldedAddressValue

Description: A type of Address that has a discrete field and value for each type of boundary or identifier down to the lowest level of detail. For example "street number" is one field, "street name" is another field, etc. Reference: MEF 79 (Sn 8.9.2)

Name	Type	Multiplicity	Description
city*	string	1	The city that the address is in
country*	string	1	Country that the address is in
geographicSubAddress	GeographicSubAddress	0..1	Additional fields used to specify an address, as detailed as possible.
locality	string	0..1	The locality that the address is in
postcode	string	0..1	Descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as zip code)
postcodeExtension	string	0..1	An extension of a postal code. E.g. the part following the dash in a US urban property address
stateOrProvince	string	0..1	The State or Province that the address is in

Name	Type	Multiplicity	Description
streetName*	string	1	Name of the street or other street type
streetNr	string	0..1	Number identifying a specific property on a public street. It may be combined with streetNrLast for ranged addresses. MEF 79 defines it as required however as in certain countries it is not used we make it optional in API.
streetNrLast	string	0..1	Last number in a range of street numbers allocated to a property
streetNrLastSuffix	string	0..1	Last street number suffix for a ranged address
streetNrSuffix	string	0..1	The first street number suffix
streetSuffix	string	0..1	A modifier denoting a relative direction
streetType	string	0..1	The type of street (e.g., alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf)

7.2.2.4. Type FormattedAddress

Description: A type of Address that has discrete fields for each type of boundary or identifier with the exception of street and more specific location details, which are combined into a maximum of two strings based on local postal addressing conventions. Reference: MEF 79 (Sn 8.9.3)

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Multiplicity	Description
addrLine1*	string	1	The first address line in a formatted address
addrLine2	string	0..1	The second address line in a formatted address
city*	string	1	The city that the address is in
country*	string	1	Country that the address is in
locality	string	0..1	An area of defined or undefined boundaries within a local authority or other legislatively defined area, usually rural or semi-rural in nature
postcode	string	0..1	Descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as ZIP code)
postcodeExtension	string	0..1	An extension of a postal code. E.g. the part following the dash in an US urban property address
stateOrProvince	string	0..1	The State or Province that the address is in

7.2.2.5. Type GeographicPoint

Description: A GeographicPoint defines a geographic point through coordinates.
Reference: MEF 79 (Sn 8.9.5)

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Multiplicity	Description
spatialRef*	string	1	The spatial reference system used to determine the coordinates (e.g. "WGS84"). The system used and the value of this field are to be agreed during the onboarding process.
x*	string	1	The latitude expressed in the format specified by the `spacialRef`
y*	string	1	The longitude expressed in the format specified by the `spacialRef`
z	string	0..1	The elevation expressed in the format specified by the `spacialRef`

7.2.2.6. Type GeographicAddressLabel

Description: A unique identifier controlled by a generally accepted independent administrative authority that specifies a fixed geographical location. Reference: MEF 79 (Sn 8.9.4)

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Multiplicity	Description
externalReferenceId*	string	1	A reference to an address by id
externalReferenceType*	string	1	Uniquely identifies the authority that specifies the addresses reference and/or its type (if the authority specifies more than one type of address). The value(s) to be used are to be agreed during the onboarding. For North American providers this would normally be CLLI (Common Language Location Identifier) code.

7.2.2.7. Type GeographicSubAddress

Description: Additional fields used to specify an address, as detailed as possible.

Name	Type	Multiplicity	Description
buildingName	string	0..1	Allows for identification of places that require building name as part of addressing information
levelNumber	string	0..1	Used where a level type may be repeated e.g. BASEMENT 1, BASEMENT 2
levelType	string	0..1	Describes level types within a building

Name	Type	Multiplicity	Description
privateStreetName	string	0..1	"Private streets internal to a property (e.g. a university) may have internal names that are not recorded by the land title office
privateStreetNumber	string	0..1	Private streets numbers internal to a private street
subUnit	GeographicSubAddressUnit[]	0..*	Representation of a MEFSubUnit It is used for describing subunit within a subAddress e.g. BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.

7.2.2.8. Type GeographicSubAddressUnit

Description: Allows for sub unit identification

Name	Type	Multiplicity	Description
subUnitNumber*	string	1	The discriminator used for the subunit, often just a simple number but may also be a range.
subUnitType*	string	1	The type of subunit e.g.BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.

7.2.2.9. Type GeographicAddressRef

Description: A reference to a Geographic Address resource available through Address Validation API.

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Multiplicity	Description

Name	Type	Multiplicity	Description
href	string	0..1	Hyperlink to the referenced GeographicAddress. Hyperlink MAY be provided by the SOF in responses. Hyperlink MUST be ignored by the SOF in case it is provided by the BA in a request
id*	string	1	Identifier of the referenced Geographic Address. This identifier is assigned during a successful address validation request (Geographic Address Validation API)

7.2.2.10. Type GeographicSiteRef

Description: A reference to a Geographic Site resource available through Service Site API

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Multiplicity	Description
href	string	0..1	Hyperlink to the referenced GeographicSite. Hyperlink MAY be provided by the SOF in responses. Hyperlink MUST be ignored by the SOF in case it is provided by the BA in a request
id*	string	1	Identifier of the referenced Geographic Site.

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 EventSubscriptionInput

Description: This class is used to register for Notifications.

Name	Type	Multiplicity	Description
callback*	string	1	This callback value must be set to *host* property from BUS Servic (serviceInventoryNotification.api.yaml). This property is appended v specified in that API to construct an URL to which notification is ser "https://bus.com/listenerEndpoint", the service state change event no `https://bus.com/listenerEndpoint/mefApi/legato/serviceInventoryNc

Name	Type	Multiplicity	Description
query	string	0..1	This attribute is used to define to which type of events to register to. serviceStateChangeEvent". To subscribe for more than one event typ `eventType=serviceCreateEvent,serviceStateChangeEvent`. The pos 'ServiceEventType' in serviceInventoryNotification.api.yaml. An em ending in subscription for all event types.

7.2.3.2. Type EventSubscription

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

Name	Type	Multiplicity	Description
callback*	string	1	The value provided by in `EventSubscriptionInput` during notification registration
id*	string	1	An identifier of this Event Subscription assigned when a resource is created.
query	string	0..1	The value provided by the `EventSubscriptionInput` during notification registration

7.2.4. Common

Types described in this subsection are shared among two or more LSO APIs.

7.2.4.1. enum BusSofType

Description: An enumeration with BUS and SOF values.

Value	MEF 99	Description
bus	BUS	
sof	SOF	

7.2.4.2. Type Note_BusSof

Description: Extra information about a given entity. Only useful in processes involving human interaction. Not applicable for an automated process.

Name	Type	Multiplicity	Description
author*	string	1	Author of the note
date*	date-time	1	Date of the note

Name	Type	Multiplicity	Description
id*	string	1	Identifier of the note within its containing entity (may or may not be globally unique, depending on provider implementation)
source*	BusSofType	1	Indicates if this Note was added by BUS or SOF.
text*	string	1	Text of the note

7.2.4.3. Type RelatedContactInformation

Description: Contact information of an individual or organization playing a role for this Service. The rule for mapping a represented attribute value to a `role` is to use the *lowerCamelCase* pattern.

Name	Type	Multiplicity	Description
emailAddress*	string	1	Email address
name*	string	1	Name of the contact
number*	string	1	Phone number
numberExtension	string	0..1	Phone number extension
organization	string	0..1	The organization or company that the contact belongs to
postalAddress	FieldedAddressValue	0..1	Identifies the postal address of the person or office to be contacted.
role*	string	1	A role the party plays in a given context.

The `role` attribute is used to provide a reason the particular party information is used. It can result from business requirements (e.g. SOF Contact Information) or from the Service Specification requirements.

The rule for mapping a represented attribute value to a `role` is to use the *lowerCamelCase* pattern e.g.

- BUS Contact: `role` equal to `busInformation`
- SOF Contact: `role` equal to `sofContact`

7.3. Notification API Data model

Figure 17 presents the Service Inventory Notification data model.

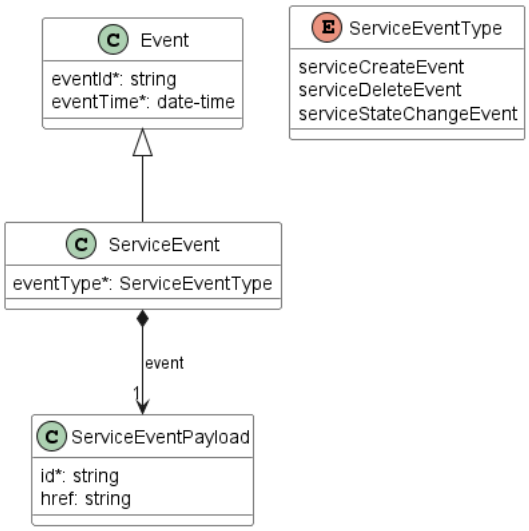

Figure 17. Service Inventory Notification Data Model

This data model is used to construct requests and responses of the API endpoints described in Section 5.2.2.

7.3.1. Type Event

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

Name	Type	Multiplicity	Description
eventId*	string	1	Id of the event
eventTime*	date-time	1	Date-time when the event occurred

7.3.2. Type ServiceEvent

Description:

Inherits from:

- Event

Name	Type	Multiplicity	Description
eventType*	ServiceEventType	1	Indicates the type of the event.
event*	ServiceEventPayload	1	A reference to the Service that is source of the notification.

7.3.3. Type ServiceEventPayload

Description: The identifier of the Service being subject of this event.

Name	Type	Multiplicity	Description

Name	Type	Multiplicity	Description
id*	string	1	ID of the Service
href	string	0..1	Hyperlink to access the Service

7.3.4. enum ServiceEventType

Description: Indicates the type of Service event.

Value
serviceCreateEvent
serviceDeleteEvent
serviceStateChangeEvent
serviceAttributeValueChangeEvent

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 10.4, Subscriber Ethernet Services Attributes, December 2018
- MEF 26.2, External Network Network Interface (ENNI) and Operator Service Attributes, August 2016
- MEF 55.1 Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, February 2021
- MEF 61.1, IP Service Attributes, May 2019
- MEF 61.1.1, Amendment to MEF 61.1: UNI Access Link Trunks, IP Addresses, and Mean Time to Repair Performance Metric, July 2022
- MEF 70, SD-WAN Service Attributes and Services, July 2019
- MEF 79, Address, Service Site, and Product Offering Qualification Management, Requirements and Use Cases, November 2019
- MEF 79.0.1, Amendment to MEF 79: Address, Service Site, and Product Offering Qualification Management, Requirements, and Use Cases, December 2020
- MEF 79.0.2, Amendment to MEF 79: Address Validation, July 2021
- [MEF W100], LSO Legato Service Specification - SD-WAN Schema Guide
- [MEF W101], LSO Legato Service Specification - Carrier Ethernet Schema Guide
- [MEF W102], LSO Legato Service Specification - IP/IP-VPN Schema Guide
- MEF 121, LSO Cantata and LSO Sonata Address Management API - Developer Guide, May 2022
- MEF 122, LSO Cantata and LSO Sonata Site Management API - Developer Guide, May 2022
- MEF 128, LSO API Security Profile, July 2022
- RFC2119, Key words for use in RFCs to Indicate Requirement Levels, by S. Bradner, March 1997
- RFC3986 Uniform Resource Identifier (URI): Generic Syntax, January 2005
- RFC8174, Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, by B. Leiba, May 2017, Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.
- RFC7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, June 2014 https://tools.ietf.org/html/rfc7231
- TMF630 TMF630 API Design Guidelines 4.2.0
- TMF638 TMF638 Service Inventory API User Guide, May 2020

Appendix A Acknowledgments

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

Mike **BENCHECK**

Michał **ŁĄCZYŃSKI**

Jack **PUGACZEWSKI**

Karthik **SETHURAMAN**

Mehmet **TOY**