



Working Draft
MEF W142 v0.1

LSO Cantata and LSO Sonata Product Catalog API - Developer Guide

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

February 2023

EXPORT CONTROL: This document contains technical data. The download, export, re-export or disclosure of the technical data contained in this document may be restricted by applicable U.S. or foreign export laws, regulations and rules and/or applicable U.S. or foreign sanctions ("Export Control Laws or Sanctions"). You agree that you are solely responsible for determining whether any Export Control Laws or Sanctions may apply to your download, export, reexport or disclosure of this document, and for obtaining (if available) any required U.S. or foreign export or reexport licenses and/or other required authorizations.

Disclaimer

© MEF Forum 2023. All Rights Reserved.

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

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

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

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

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

Copyright

© MEF Forum 2023. Any reproduction of this document, or any portion thereof, shall contain the following statement: "Reproduced with permission of MEF Forum." No user of this document is authorized to modify any of the information contained herein.

Table of Contents

- List of Contributing Members
- 1. Abstract
- 2. Terminology and Abbreviations
- 3. Compliance Levels
- 4. Introduction
 - 4.1. Conventions in the Document
 - 4.2. Relation to Other Documents
 - 4.3. Approach
 - 4.4. General concept
 - 4.4.1. General concept introduction
 - 4.4.2 JSON Subschema
 - 4.4.3 Product Specification and Product Offering Schemas
 - 4.5. High-Level Flow
- 5. API Description
 - 5.1. High-level use cases
 - 5.2. API Endpoint and Operation Description
 - 5.2.1. Seller side API Endpoints
 - 5.2.2. Buyer side API Endpoints
 - 5.3. Specifying the Buyer ID and the Seller ID
 - 5.4. Model Structural Validation
 - 5.5. Security Considerations
- 6. API Interactions and Flows
 - 6.1. Product Category Use Cases
 - 6.1.1 Product Category - Model
 - 6.1.2 Product Category - Lifecycle
 - 6.1.3 Use case 1: Retrieve Product Category List
 - 6.1.3.1 Interaction flow
 - 6.1.3.2. Retrieve Product Category List - Request
 - 6.1.3.3. Retrieve Product Category List - Response
 - 6.1.4 Use case 2: Retrieve Product Category by Identifier
 - 6.1.4.1 Interaction flow
 - 6.1.4.2. Retrieve Product Category by Identifier - Request
 - 6.1.4.3. Retrieve Product Category by Identifier - Response
 - 6.2. Product Offering Use Cases
 - 6.2.1 Product Offering - Model
 - 6.2.1.1 Introduction to the model
 - 6.2.1.2 Product Offering Specification Schema
 - 6.2.1.3 Product Offering Contextual Info
 - 6.2.2 Product Offering - Lifecycle
 - 6.2.3 Use case 3: Retrieve Product Offering List

- 6.2.3.1 Interaction flow
 - 6.2.3.2. Retrieve Product Offering List - Request
 - 6.2.3.3. Retrieve Product Offering List - Response
 - 6.2.4 Use case 4: Retrieve Product Offering by Identifier
 - 6.2.4.1 Interaction flow
 - 6.2.4.2. Retrieve Product Offering by Identifier - Request
 - 6.2.4.3. Retrieve Product Offering by Identifier - Response
 - 6.3. Product Specification Use Cases
 - 6.3.1 Product Specification - Model
 - 6.3.2 Product Specification - Lifecycle
 - 6.3.3 Use case 5: Retrieve Product Specification List
 - 6.3.3.1 Interaction flow
 - 6.3.3.2. Retrieve Product Specification List - Request
 - 6.3.3.3. Retrieve Product Specification List - Response
 - 6.3.4 Use case 6: Retrieve Product Specification by Identifier
 - 6.3.4.1 Interaction flow
 - 6.3.4.2. Retrieve Product Specification by Identifier - Request
 - 6.3.4.3. Retrieve Product Specification by Identifier - Response
 - 6.9. Use case 7: Register for Event Notifications
 - 6.9.1. Register for Event Notifications - Request
 - 6.9.2. Register for Event Notifications - Response
 - 6.9.3. Unregister for Event Notifications - Request
 - 6.9.4. Unregister for Event Notifications - Response
 - 6.10. Use case 8: Send Event Notification
- 7. API Details
 - 7.1. API patterns
 - 7.1.1. Indicating errors
 - 7.1.1.1. Type Error
 - 7.1.1.2. Type Error400
 - 7.1.1.3. `enum` Error400Code
 - 7.1.1.4. Type Error401
 - 7.1.1.5. `enum` Error401Code
 - 7.1.1.6. Type Error403
 - 7.1.1.7. `enum` Error403Code
 - 7.1.1.8. Type Error404
 - 7.1.1.9. Type Error500
 - 7.1.1.10. Type Error501
 - 7.1.2. Response pagination
 - 7.2. API Data model
 - 7.2.1 Product Category
 - 7.2.1.1 Type ProductCategory
 - 7.2.1.2 `enum` CategoryLifecycleStatusType

- 7.2.1.3 Type CategoryRef
 - 7.2.1.4 Type ProductOfferingRef
- 7.2.2 Product Offering
 - 7.2.2.1 Type ProductOffering_Common
 - 7.2.2.2 Type ProductOffering
 - 7.2.2.3 Type ProductOffering_Find
 - 7.2.2.4 **enum** ProductOfferingLifecycleStatusType
 - 7.2.2.5 Type ProductOfferingLifecycleStatusTransition
 - 7.2.2.6 Type ProductSpecificationRef
 - 7.2.2.7 Type MEFItemTerm
 - 7.2.2.8 **enum** MEFEndOfTermAction
 - 7.2.2.9 Type ProductOfferingContextualInfo
 - 7.2.2.10 Type Context
 - 7.2.2.11 **enum** MEFProductAction
 - 7.2.2.12 **enum** MEFBusinessFunction
 - 7.2.2.13 Type Region
- 7.2.3 Product Specification
 - 7.2.3.1 Type ProductSpecification_Common
 - 7.2.3.2 Type ProductSpecification
 - 7.2.3.3 Type ProductSpecification_Find
 - 7.2.3.4 **enum** ProductSpecificationLifecycleStatusType
 - 7.2.3.5 Type ProductSpecificationRelationship
- 7.2.4 Common types
 - 7.2.4.1 Type AttachmentValue
 - 7.2.4.2 Type MEFByteSize
 - 7.2.4.3 **enum** DataSizeUnit
 - 7.2.4.4 **enum** MEFBuyerSellerType
 - 7.2.4.5 Type RelatedContactInformation
 - 7.2.4.6 Type FieldedAddress
 - 7.2.4.7 Type GeographicSubAddress
 - 7.2.4.8 Type MEFSubUnit
 - 7.2.4.9 Type Note
 - 7.2.4.10 Type SchemaRefOrValue
 - 7.2.4.11 Type Duration
 - 7.2.4.12 **enum** TimeUnit
- 7.2.5 Notification Registration
- 7.2.8.1. Type EventSubscriptionInput
 - 7.2.8.2. Type EventSubscription
- 7.3. Notification API Data model
 - 7.3.1. Type Event
 - 7.3.2. Type ProductCategoryEvent
 - 7.3.3. Type ProductCategoryEventPayload

- 7.3.4. `enum` ProductCategoryEventType
- 7.3.5. Type ProductOfferingEvent
- 7.3.6. Type ProductOfferingEventPayload
- 7.3.7. `enum` ProductOfferingEventType
- 7.3.8. Type ProductSpecificationEvent
- 7.3.9. Type ProductSpecificationEventPayload
- 7.3.10. `enum` ProductSpecificationEventType
- 8. References

List of Contributing Members

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

Member

Table 1. Contributing Members

1. Abstract

This standard is intended to assist implementation of the Product Catalog functionality defined for the LSO Cantata and LSO Sonata Interface Reference Points (IRPs), for which requirements and use cases are defined in MEF 127 *Product Catalog Requirements and Use Cases* [MEF127]. This standard consists of this document and complementary API definitions for Product Catalog Querying and Product Catalog Notifications.

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

<https://github.com/MEF-GIT/MEF-LSO-Sonata-SDK>

- `productApi/catalog/productCatalog.api.yaml`
- `productApi/catalog/productCatalogNotification.api.yaml`

<https://github.com/MEF-GIT/MEF-LSO-Cantata-SDK>

- `productApi/catalog/productCatalog.api.yaml`
- `productApi/catalog/productCatalogNotification.api.yaml`

The Product Catalog API is defined using OpenAPI 3.0 [OAS-V3]

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 standards referenced below are included in this document by reference and are not repeated in the table below:

- MEF 55.1.1 Lifecycle Service Orchestration (LSO): Reference Architecture and Framework [[MEF 55.1.1](#)]
- MEF 57.2 Product Order Management Requirements and Use Cases [[MEF57.2](#)]
- MEF 127 Product Catalog Requirements and Use Cases [[MEF127](#)]
- MEF 79 Address, Service Site, and Product Offering Qualification Management, Requirements and Use Cases, November 2019 [[MEF79](#)]

Term Description Reference JSON subschema JSON schema A is called the subschema of schema B when every JSON that is valid against schema A is valid against schema B. This document (Chapter 4.4.2)

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 Product Catalog API allows the Buyer to retrieve Product Specifications, Product Offerings, and Product Categories they are assigned to. The API defines notifications related to the lifecycle of these entities. This allows the establishment of commercial synchronization between the Seller and potential Buyers by the possibility of zero-touch introduction of the new Product Specifications and Product Offerings to the market.

This standard specifies the Application Programming Interface (API) for Product Catalog functionality of the LSO Cantata IRP and LSO Sonata IRP as defined in the *MEF 55.1 Lifecycle Service Orchestration (LSO): Reference Architecture and Framework* [MEF55.1]. The LSO Reference Architecture is shown in Figure 1 with both IRPs highlighted.

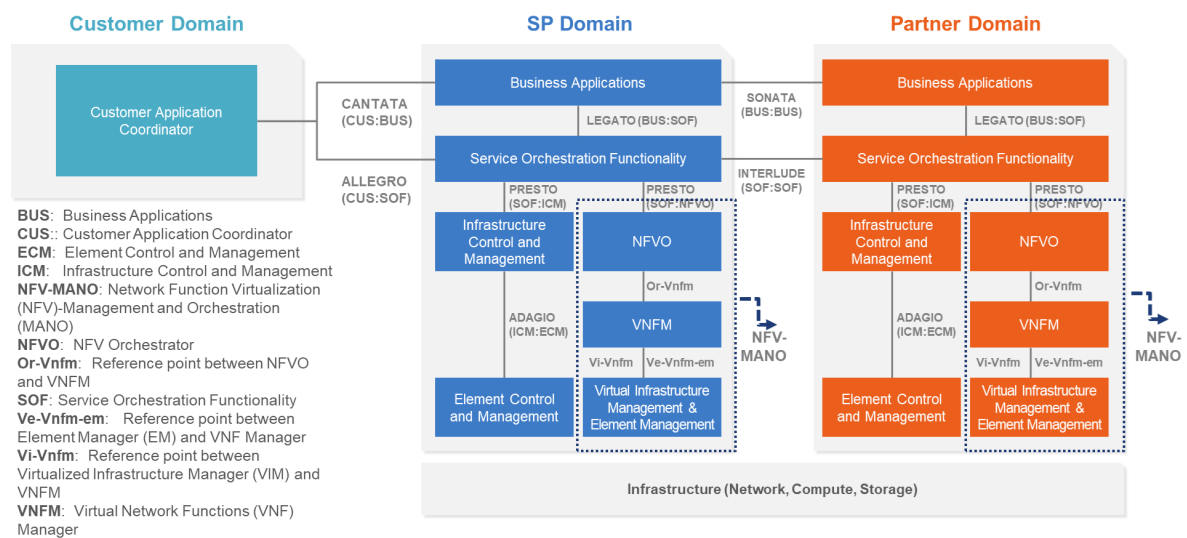


Figure 1. The LSO Reference Architecture

Cantata and Sonata IRPs define pre-ordering and ordering functionalities that allow an automated exchange of information between business applications of the Buyer (Customer or Service Provider) and Seller (Service Provider or Partner) Domains. Those are:

- Product Catalog
- Address Validation
- Site Retrieval
- Product Offering Qualification
- Product Quote
- Product Inventory
- Product Ordering
- Trouble Ticketing
- Billing

The business requirements and use cases for Product Catalog are defined in MEF 127 *Product Catalog Requirements and Use Cases* [MEF127].

Note: [TMF620] Product Catalog API covers use cases related to Product Catalog management as well. Whereas the goal of this API (specified in this document) is to allow the publishing of Product Offerings and Product Specifications and onboarding of them (by the Buyer) in a fast and efficient way (inter-carrier read-only API).

This document is structured as follows:

- [Chapter 4](#) provides an introduction to Product Catalog and its description in a broader context of Cantata and Sonata and their corresponding SDKs.
- [Chapter 5](#) gives an overview of endpoints, resource model and design patterns.
- Use cases and flows are presented in [Chapter 6](#).
- And finally, [Chapter 7](#) complements previous sections with a detailed API description.

4.1. 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. `ProductOffering`).
- 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 ``` notation is used to indicate a variable to be substituted with a correct value.

4.2. Relation to Other Documents

This API implements the Product Catalog related requirements and use cases that are defined in MEF 127 [MEF127]. The API definition builds on *TMF620 Product Catalog API REST Specification 4.1.0* [TMF620].

4.3. Approach

As presented in Figure 2, both Cantata and Sonata API frameworks consist of three structural components:

- Generic API framework
- Product-independent information (Function-specific information and Function-specific operations)

- Product-specific information (MEF product specification data model)

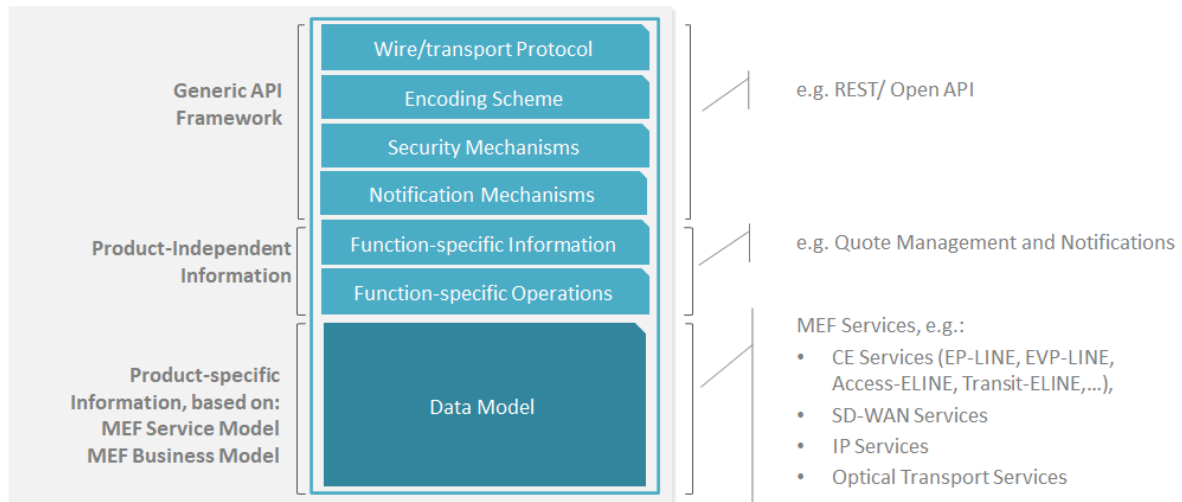


Figure 2. Cantata and Sonata API framework

The essential concept behind the framework is to decouple the common structure, information and operations from the specific product information content.

Firstly, the Generic API Framework defines a set of design rules and patterns that are applied across all Cantata or Sonata APIs.

Secondly, the product-independent information of the framework focuses on a model of a particular Cantata or Sonata functionality and is agnostic to any of the product specifications.

Finally, the product-specific information part of the framework focuses on MEF product specifications that define business-relevant attributes and requirements for trading MEF subscriber and MEF operator services.

In this framework, Product Catalog is a central system that hosts Product-specific definitions. Every product-related operation starts in Product Catalog, firstly by defining the product then each actor in the flow reads and interprets the retrieved product to use it for the purposes defined by its responsibility.

4.4. General concept

4.4.1. General concept introduction

Product Catalog introduces three main types: Product Specification, Product Offering, and Product Category. Modeling starts with the introduction of the Product Specification type which defines the Product's attributes (with their types, cardinalities, and allowable values) and relationships that defines how the Product may be related to other Products. Once Product Specifications are defined, they should be available for marketing purposes. Because terms and conditions which define whether a Product is available on the market are usually organized by different business processes, the Product Offering type has been

introduced. Product Offering may further constrain or concretize the attributes values, and cardinalities of the relations to define the desired offer with a settled price.

In short, Product Specification is the definition of the Product, and Product Offering is its exposition on the market.

Product Specification may be exposed on the market by many Product Offerings, at the same time Product Offerings based on the different Product Specifications may be related e.g. by the technology variant. To organize the catalog (by organizing loosely defined Product Offerings) Product Categories have been introduced. Product Category allows grouping of Product Offerings and names this group by its purpose. To make the categorizing reusable it is allowed to group Product Categories as well.

4.4.2 JSON Subschema

JSON Subschema term has been introduced to allow defining specialization of particular JSON Schema. Specialization should be understood as narrowing the set of JSONs that are valid against the given JSON Schema.

JSON Subschema definition is formalized as:

JSON schema A is called the subschema of schema B when every JSON that is valid against schema A is valid against schema B.

The rules that are used to constrain or restrict the JSON Schema are described in the chapter [[Product Offering Specification Schema](#)].

Let's consider exemplary JSON Schema (this is the reduced JSON Schema of AccessElineOvc for exemplary purposes):

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://mef.com/product.schema.json",
  "title": "AccessElineOvc",
  "type": "object",
  "properties": {
    "maximumFrameSize": {
      "type": "integer",
      "minimum": 1526
    },
    "ceVlanIdPreservation": {
      "type": "string",
      "enum": [
        "PRESERVE",
        "STRIP",
        "RETAIN"
      ]
    }
  }
}
```

JSON Subschema of the schema above could be constructed as:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://mef.com/product.schema.json",
  "title": "AccessELineOvc",
  "type": "object",
  "properties": {
    "maximumFrameSize": {
      "type": "integer",
      "minimum": 1526
    },
    "ceVlanIdPreservation": {
      "type": "string",
      "enum": [
        "PRESERVE",
        "STRIP"
      ]
    }
  },
  "required": ["maximumFrameSize"]
}
```

Considering two following JSON payloads:

- Payload A:

```
{
  "maximumFrameSize": 1526,
  "ceVlanIdPreservation" : "STRIP"
}
```

- Payload B:

```
{
  "maximumFrameSize": 1526,
  "ceVlanIdPreservation" : "RETAIN"
}
```

we see that Payload A can is compliant with both schemas. Payload B is valid only against the first schema. Thus schema two, as more restrictive, is a subschema of the first schema.

4.4.3 Product Specification and Product Offering Schemas

The chapters above introduced the Product Specification as the type which defines the attributes of the Product. Those attributes are organized as [JSON Schema] which is called

Source Schema.

Product Offering as the type which exposes Product Specification to the market may constraint the attributes defined in the Source Schema which produces another JSON Schema specific for the Product Offering which is called the Intermediate Schema. This schema is the JSON Sub-schema of the Source Schema.

Additionally, it may be required to constrain the Intermediate Schema in the context of a particular Business Function and Product Action (e.g. some attributes are not relevant for the Product Offering Qualification business function). It is possible to define such cases by

Contextual Schema for a given Business Function and Product Action context. This schema is the JSON Sub-schema of the **Intermediate Schema**.

JSON Subschema is still JSON Schema.

Summarizing, three types of schemas may be distinguished:

- **Source Schema** which is the definition of the Product itself,
- **Intermediate Schema** which is constraint usage of the **Source Schema** for Product Offering purposes [**Intermediate Schema**],
- **Contextual Schema** which defines the contextual usage of the **Intermediate Schema** in the given context of Business Function and Product Action [**Contextual Schema**].

The below diagram depicts the above summary in graphical form.

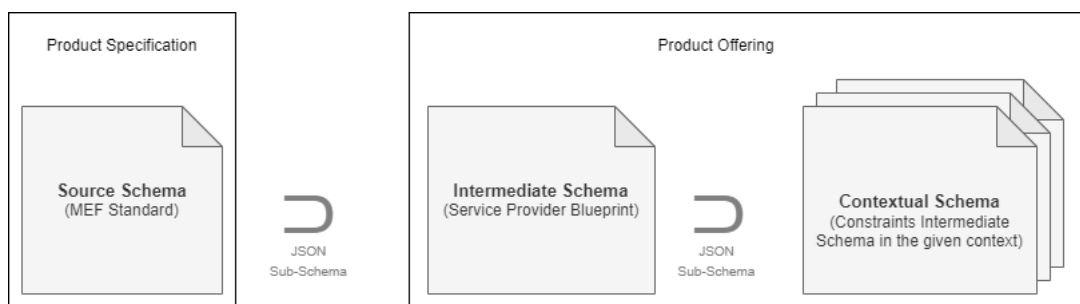


Figure 3. Relation between Source Schema, Intermediate Schema and Contextual Schema

4.5. High-Level Flow

Product Catalog is part of a broader Cantata and Sonata End-to-End flow. Figure 4 shows a high-level diagram to get a good understanding of the whole process and Product Catalog's position within it.

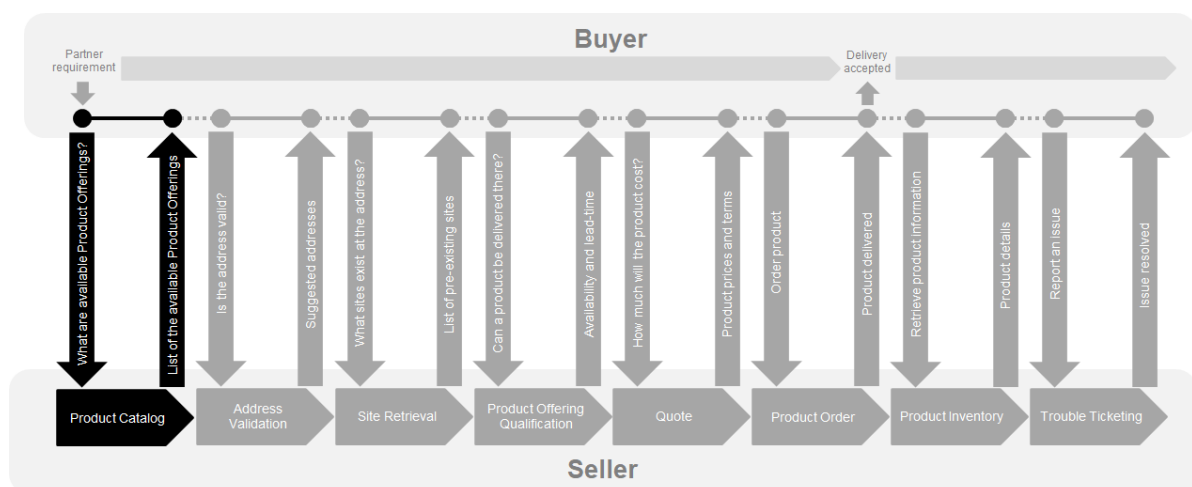


Figure 4. Cantata and Sonata End-to-End Function Flow

- Product Catalog:
 - Allows the Buyer to retrieve Product Offerings and Product Specifications describing the Products available for ordering.
- Address Validation:
 - Allows the Buyer to retrieve address information from the Seller, including exact formats, for addresses known to the Seller.
- Site Retrieval:
 - Allows the Buyer to retrieve Geographic Site information including exact formats for Geographic Sites known to the Seller.
- Product Offering Qualification (POQ):
 - Allows the Buyer to check whether the Seller can deliver a product or set of products from among their product offerings at the geographic address or a Geographic Site specified by the Buyer; or modify a previously purchased product.
- Quote:
 - Allows the Buyer to submit a request to find out how much the installation of an instance of a Product Offering, an update to an existing Product, or a disconnect of an existing Product will cost.
- Product Order:
 - Allows the Buyer to request the Seller to initiate and complete the fulfillment process of an installation of a Product Offering, an update to an existing Product, or a disconnect of an existing Product at the address defined by the Buyer.
- Product Inventory:
 - Allows the Buyer to retrieve the information about the existing Product instances from Seller's Product Inventory.
- Trouble Ticketing:
 - Allows the Buyer to create, retrieve, and update Trouble Tickets as well as receive notifications about Incidents' and Trouble Tickets' updates. This allows managing issues and situations for a Product provided by the Seller.
- Billing:
 - Allows the Buyer to retrieve the Billing (Invoice) information, download it as a document, and receive notifications of document creation.

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 gives an overview of the API resource model.

5.1. High-level use cases

Figure 5 presents a high-level use case diagram as specified in MEF 127 [[MEF127](#)] in section 8. This picture aims to help understand the endpoint mapping. Use cases are described extensively in [chapter 6](#).

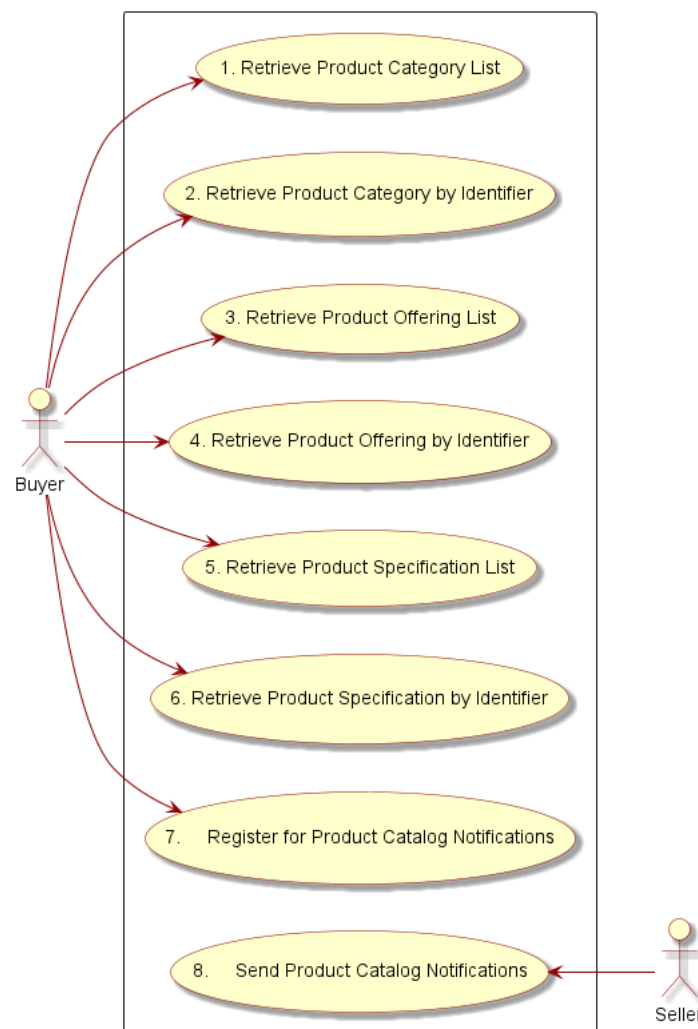


Figure 5: Use cases

5.2. API Endpoint and Operation Description

5.2.1. Seller side API Endpoints

Base URL for Cantata: `https://{serverBase}:{port}/{seller_prefix}/mefApi/cantata/productCatalog/v1/`

Base URL for Sonata: `https://{serverBase}:{port}/{seller_prefix}/mefApi/sonata/productCatalog/v1/`

The following API endpoints are implemented by the Seller and allow the Buyer to retrieve Product Categories, Product Offerings, and Product Specifications. As well as register for Notifications. The endpoints and corresponding data model are defined in

`productApi/productCatalog/productCatalog.api.yaml`.

API endpoint	Description	MEF 127 Use Case mapping
<code>GET /productCategory</code>	The Buyer requests a list of Product Categories from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Product Categories.	UC 1: Retrieve Product Category List
<code>GET /productCategory/{id}</code>	The Buyer requests detailed information about a single Product Category based on a Product Category Identifier.	UC 2: Retrieve Product Category by Identifier
<code>GET /productOffering</code>	The Buyer requests a list of Product Offerings from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Product Offerings.	UC 3: Retrieve Product Offering List
<code>GET /productOffering/{id}</code>	The Buyer requests detailed information about a single Product Offering based on a Product Offering Identifier.	UC 4: Retrieve Product Offering by Identifier
<code>GET /productSpecification</code>	The Buyer requests a list of Product Specifications from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Product Specifications.	UC 5: Retrieve Product Specification List
<code>GET /productSpecification/{id}</code>	The Buyer requests detailed information about a single Product Specification based on a Product Specification Identifier.	UC 6: Retrieve Product Specification by Identifier

Table 2. Seller side mandatory API endpoints

[R1] The Seller **MUST** implement all API endpoints listed in Table 2. [MEF127 R16]

API endpoint	Description	MEF 127 Use Case mapping
POST /hub	The Buyer requests to subscribe to Product Catalog notifications.	UC 7: Register for Product Catalog Notifications
GET /hub/{id}	A request initiated by the Buyer to retrieve the details of the notification subscription with given Identifier.	UC 7: Register for Product Catalog Notifications
DELETE /hub/	A request initiated by the Buyer to instruct the Seller to stop sending notifications.	UC 7: Register for Product Catalog Notifications

Table 3. Seller side optional API endpoints

[O1] The Seller **MAY** implement API endpoints listed in Table 3. [MEF127 O3]

[CR1]<[O1] If any of the endpoints defined in Table 3 is implemented, then The Seller **MUST** return 501 HTTP code for not implemented endpoints.

[CR2]<[O1] If any of endpoints defined in table 3 is implemented, then all of the endpoints listed in Table 3 **MUST** be implemented

[CR3]<[O1] The Seller **MUST** support at least one group of Notification Types. Where the group of Notification Types is understood as all events related to the Product Offering or Product Specification or Product Category. [MEF127 C01]

[CR4]<[O1] The Seller **MUST** support sending notifications for all notification types that are available for the listener registration. The notifications **MUST** be send to Buyer's API endpoint provided in the registration [MEF127] [MEF127 C02, MEF127 CO3, MEF127 CO4]

5.2.2. Buyer side API Endpoints

Base URL for Cantata: <https://mefApi/cantata/productCatalogNotification/v1/>

Base URL for Sonata: <https://mefApi/cantata/productCatalogNotification/v1/>

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

[productApi/catalog/productCatalogNotification.api.yaml](#)

API Endpoint	Description	MEF 127 Use Case Mapping
POST /listener/productCategoryCreateEvent	A request initiated by the Seller to notify the Buyer on new ProductCategory creation.	UC 08: Send Product Catalog Notification
POST /listener/productCategoryAttributeValueChangeEvent	A request initiated by the Seller to notify the Buyer on ProductCategory attribute value change.	UC 08: Send Product Catalog Notification
POST /listener/productCategoryStatusChangeEvent	A request initiated by the Seller to notify the Buyer on ProductCategory.lifecycleStatus status change.	UC 08: Send Product Catalog Notification
POST /listener/productOfferingCreateEvent	A request initiated by the Seller to notify the Buyer on new ProductOffering creation.	UC 08: Send Product Catalog Notification
POST /listener/productOfferingAttributeValueChangeEvent	A request initiated by the Seller to notify the Buyer on ProductOffering attribute value change.	UC 08: Send Product Catalog Notification
POST /listener/productOfferingStatusChangeEvent	A request initiated by the Seller to notify the Buyer on ProductOffering.lifecycleStatus status change.	UC 08: Send Product Catalog Notification
POST /listener/productSpecificationCreateEvent	A request initiated by the Seller to notify the Buyer on new ProductSpecification creation.	UC 08: Send Product Catalog Notification

API Endpoint	Description	MEF 127 Use Case Mapping
POST <code>/listener/productSpecificationAttributeValueChangeEvent</code>	A request initiated by the Seller to notify the Buyer on <code>ProductSpecification</code> attribute value change.	UC 08: Send Product Catalog Notification
POST <code>/listener/productSpecificationStatusChangeEvent</code>	A request initiated by the Seller to notify the Buyer on <code>ProductSpecification.lifecycleStatus</code> status change.	UC 08: Send Product Catalog Notification

Table 4. Buyer side optional API endpoints

[O2] The Buyer **MAY** support API endpoints listed in Table 4. [MEF127 O3]

[CR5]<([O1], [O2]) If endpoints listed in Table 3 are supported (for any of Notification Type) and the Buyer registered a subscription for any Notification Type then the Buyer **MUST** support corresponding endpoints from Table 4.

5.3. Specifying the Buyer ID and the Seller ID

A business entity willing to represent multiple Buyers or multiple Sellers must follow requirements of MEF 79.1 [MEF79.1] chapter 8.8, which states:

For requests of all types, there is a business entity that is initiating an Operation (called a Requesting Entity) and a business entity that is responding to this request (called the Responding Entity). In the simplest case, the Requesting Entity is the Buyer and the Responding Entity is the Seller. However, in some cases, the Requesting Entity may represent more than one Buyer and similarly, the Responding Entity may represent more than one Seller.

While it is outside the scope of this specification, it is assumed that the Requesting Entity and the Responding Entity are aware of each other and can authenticate requests initiated by the other party. It is further assumed that both the Buying Entity and the Requesting Entity know:

- a) the list of Buyers the Requesting Entity represents when interacting with this Responding Entity; and
- b) the list of Sellers that this Responding Entity represents to this Requesting Entity.

In the API the `buyerId` and `sellerId` are represented as query parameters in each operation defined in `productCatalog.api.yaml` and as attributes of events as described in `productCatalogNotification.api.yaml`.

[R2] If the Requesting Entity has the authority to represent more than one Buyer the request **MUST** include `buyerId` query parameter that identifies the Buyer being represented [MEF79 R80]

[R3] If the Requesting Entity represents precisely one Buyer with the Responding Entity, the request **MUST NOT** specify the `buyerId` [MEF79 R81]

[R4] If the Responding Entity represents more than one Seller to this Buyer the request **MUST** include `sellerId` query parameter that identifies the Seller with whom this request is associated [MEF79 R82]

[R5] If the Responding Entity represents precisely one Seller to this Buyer, the request **MUST NOT** specify the `sellerId` [MEF79 R83]

[R6] If `buyerId` or `sellerId` attributes were specified in the request same attributes **MUST** be used in the notification payload.

5.4. Model Structural Validation

The structure of the HTTP payloads exchanged via Product Catalog API endpoints is defined using OpenAPI version 3.0.

[R7] Implementations **MUST** use payloads that conform to these definitions.

5.5. Security Considerations

There must be an authentication mechanism whereby a Seller can be assured who a Buyer is and vice-versa. There must also be authorization mechanisms in place to control what a particular Buyer or Seller 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. It is specified by a separate MEF Project ([[MEF128](#)]).

6. API Interactions and Flows

This section provides a detailed insight into the API functionality, use cases, and flows. It starts with Table 5 presenting a list and short description of all business use cases then presents the variants of end-to-end interaction flows, and in the following subchapters describes the API usage flow and examples for each of the use cases.

Table 5. lists the use cases supported by Product Catalog API (use case numbers as in MEF 127 for mapping):

Use Case #	Use Case Name	Use Case Description
1	Retrieve Product Category List	The Buyer requests a list of Product Categories from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Product Categories.
2	Retrieve Product Category by Product Category Identifier	The Buyer requests detailed information about a single Product Category based on a Product Category Identifier.
3	Retrieve Product Offering List	The Buyers requests a list of Product Offerings from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Product Offering.
4	Retrieve Product Category by Product Offering Identifier	The Buyer requests detailed information about a single Product Offering based on a Product Offering Identifier.
5	Retrieve Product Specification List	The Buyers requests a list of Product Specifications from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Product Specifications.
6	Retrieve Product Specification by Product Specification Identifier	The Buyer requests detailed information about a single Product Specification based on a Product Specification Identifier.
7	Register for Event Notifications	The Buyer requests to subscribe to Product Categories, Product Offerings and Product Specifications Notifications.

Use Case		
Case #	Use Case Name	Use Case Description
8	Send Event Notification	Send Event Notification The Seller sends a notification regarding a Product Category, Product Offering, or Product Specification to the Buyer.

Table 5. Use cases description

Figure 6 presents an example of the flow of `ProductOffering` lifecycle and possible related requests.

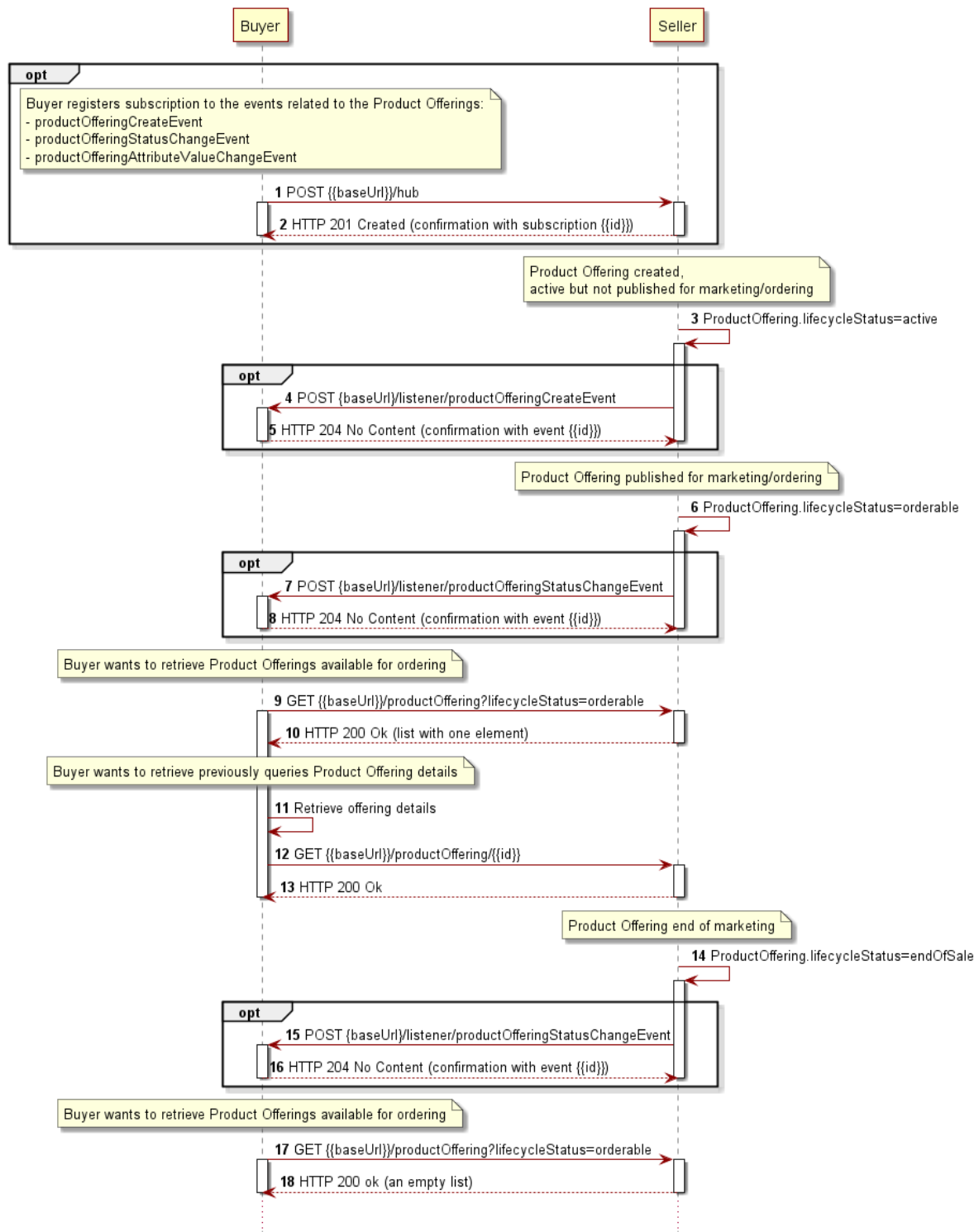


Figure 6. Exemplary API flow for Product Offering

Registration for events is optional, so all sequences on the diagram related to notification exchange were framed as optional, so as the below description of the sequence diagram.

- (optional) The Buyer registers the listener by sending the request (1) with specified `eventType` as `productOfferingCreateEvent`, `productOfferingStatusChangeEvent`, and `productOfferingAttributeValueChangeEvent`.
- (optional) (2) The Seller responds with success.
- The Seller publishes new `ProductOffering` with `lifecycleStatus=active` (3).

- (optional) What causes sending notification to the Buyer (4-5) with the type `productOfferingCreateEvent`.
- The Seller decides to make the previously created offering available for ordering by changing `lifecycleStatus` to `orderable` (6).
- (optional) What causes sending notification to the Buyer (7-8) with type the `productOfferingStatusChangeEvent`.
- The Buyer decides to ask for offerings available for ordering by sending a request (9) providing in query parameters required `lifecycleStatus` as `orderable`.
- (10) The Seller responds with the list of `orderable` offerings.
- (11) The Buyer asks for details of the retrieved offering by sending a request (12) with the specified `identifier` of the offering.
- (13) The Seller responds with detailed offering information.
- The Seller decides to end selling previously published offering by changing `lifecycleStatus` to `endOfSale` (14).
- (optional) What causes sending notification to the Buyer (15-16) with the type `productOfferingStatusChangeEvent`.
- The Buyer asks again for offerings available for ordering by sending a request (17) providing in query parameters required `lifecycleStatus` as `orderable`.
- Because there is no offering that is available for sale, the Seller responds with empty list (18).

The detailed business requirements of each of the use cases are described in section 8 of MEF 127 [MEF127].

6.1. Product Category Use Cases

6.1.1 Product Category - Model

Product Categories are designed to be used for technological segregation e.g. grouping Product Offerings delivered via Fiber medium.

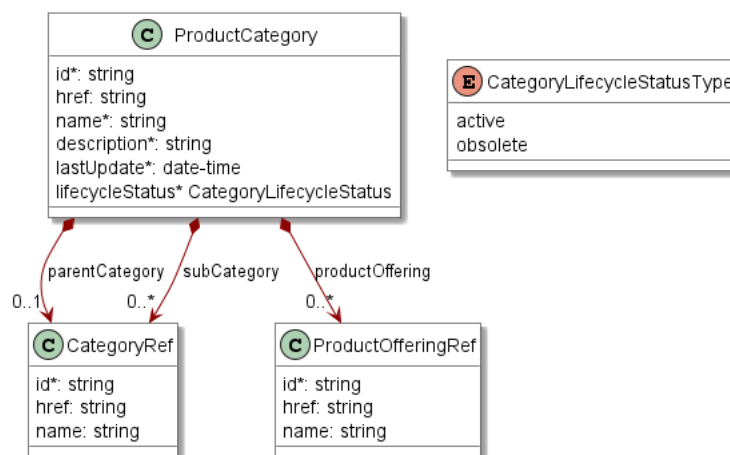


Figure 7. Product Category Model

[O3] The Seller **MAY** update the following Product Category attributes:

- `description`
- `lastUpdate`
- `lifecycleStatus`
- `parentCategory`
- `subCategory`
- `productOffering`

[R8] If the Product Category's `lifecycleStatus` reached `active` status, then the changes introduced by the Seller **MUST** respect the backward compatibility for Product Category, which is defined as:

- `description` may be changed to any other text,
- `lifecycleStatus` may be changed accordingly to the state machine,
- `parentCategory` category reference may be changed,
- `subCategory` adding and removing of category's references is allowed,
- `productOffering` adding and removing of offering's references is allowed.

[R9] After a Product Category has been created, the attributes `id` and `name` **MUST NOT** be modified. [MEF127 R22]

[R10] In the case of a change of any attribute defined in [O3], the Seller **MUST** update the `lastUpdate` attribute with the date of the most recent modification. [MEF127 R21]

[R11] If a Product Category has a parent category, then its `id` **MUST** be in the `subCategory` list of the referenced Product Category. Consequently, if a Product Category A is specified in the `subCategory` list of Product Category B, then the `id` of Product Category B **MUST** be included in the `parentCategory` attribute of Product Category A. [MEF127 R17, MEF127 R19]

[R12] If a Product Category has no parent Category, then its `id` **MUST NOT** be in the `subCategory` list for any Product Category. [MEF127 R18]

[R13] The `productOffering` attribute **MUST** contain all Product Offerings that reference this Product Category. [MEF127 R20]

6.1.2 Product Category - Lifecycle

Figure 8 presents the Product Category state machine:

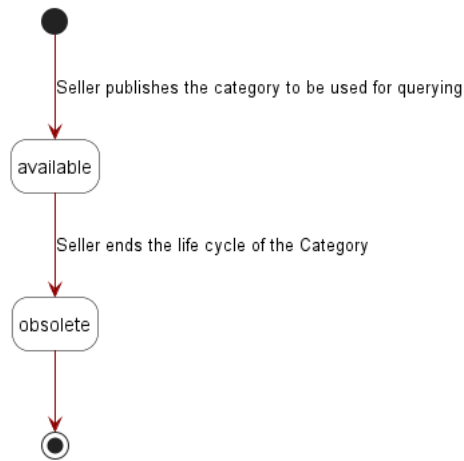


Figure 8. Product Category State Machine

The Product Category State Machine is very simple and this state machine aims to make Product Categories available for querying purposes or not.

Table 6 presents the mapping between API `lifecycleStatus` values (aligned with TMF) and MEF127 naming together with its descriptions.

lifecycleStatus	MEF 127 name	Description
<code>active</code>	AVAILABLE	The Product Category can be used by the Buyer to retrieve Product Offerings.
<code>obsolete</code>	OBSOLETE	The Product Category is <code>obsolete</code> when it can no longer be used by the Buyer to retrieve Product Offerings. The Product Category with this state may be removed from the Product Catalog. This is a final state.

Table 6. Product Category lifecycle statuses

[R14] The Seller **MUST** support all statuses for Product Category and the associated state transitions [MEF127 R65].

[O4] The Seller **MAY** update the Product Category (see chapter [6.1.1] to find updatable attributes) when the Product Category `lifecycleStatus` is equal to `active`.

[R15] The Seller **MUST NOT** update the Product Category when it is in the final status - `obsolete`.

6.1.3 Use case 1: Retrieve Product Category List

6.1.3.1 Interaction flow

The flow of this use case is very simple and is described in Figure 9.

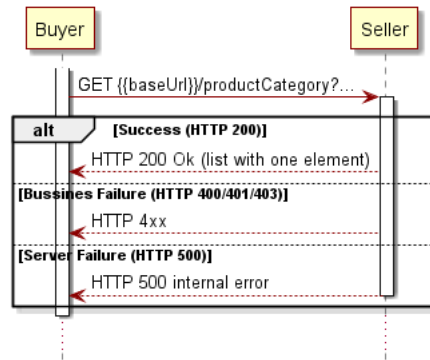


Figure 9: Use Case 1 - Retrieve Product Category List

The Buyer wants to retrieve the list of Product Categories that match the given filtering criteria. Later on, the result is used to query Product Offerings by category.

6.1.3.2. Retrieve Product Category List - Request

[O5] The Buyer **MAY** retrieve the list of Product Categories by using a `GET /productCategory` operation with desired filtering criteria. The attributes that are available to be used are **[MEF127 O4]**:

- `name`
- `lastUpdate.gt`
- `lastUpdate.lt`
- `lifecycleStatus`
- `parentCategory.id`

[CR5]<[O5] The Seller **MUST** reject the request if the attributes requested by the Buyer are not supported value defined in**[O5]**. **[MEF127 R4]**

The Buyer may also ask for pagination with the use of the `offset` and `limit` parameters. The filtering and pagination attributes must be specified in URI query format [RFC3986](#). Section [7.1.2](#). provides details about the implementation of pagination mechanism.

```
https://serverRoot/mefApi/sonata/productCatalog/v1/category?status=active&limit=10&offset=0
```

The example above shows a Buyer's request to get the first ten Product Categories that are in `active` status. The correct response (HTTP code `200`) in the response body contains a list of `ProductCategory` objects matching the criteria. To get more details (e.g. the list of Product Offerings in the given Product Category), the Buyer has to query a specific `ProductCategory` by `id`.

6.1.3.3. Retrieve Product Category List - Response

The snippet below presents an example of the Retrieve Product Category List response:

Retrieve **ProductCategory** List Response

Headers:

```
X-Result-Count=1
X-Total-Count=1
X-Pagination-Throttled=false
```

Body:

```
[
  {
    "id": "productCategory-2",
    "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productCategory/productCategory-2",
    "name": "Access E-Lines Product Offerings",
    "description": "This category groups are available Access E-Line Product Offerings",
    "lastUpdate": "2023-01-19T16:33:20.324Z",
    "lifecycleStatus": "active",
    "parentCategory": {
      "id": "productCategory-1",
      "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/category/productCategory-1"
    },
    "productOffering": [
      {
        "id": "productOffering-1",
        "href":
"http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-1"
      },
      {
        "id": "productOffering-2",
        "href":
"http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-2"
      }
    ]
  }
]
```

[R16] The Seller **MUST** put the following attributes into the **ProductCategory** object in the response [MEF127 R23, MEF127 25]:

- **id**
- **name**
- **description**
- **lastUpdate**
- **lifecycleStatus**

[R17] The Seller response **MUST** include the following attributes into the **ProductCategory** if they are set by the Seller [MEF127 R24, MEF127 R25]:

- **productOffering**
- **parentCategory**
- **subCategory**

[R18] If case no items matching the criteria are found, the Seller **MUST** return a valid response with an empty list. [MEF127 R26]

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

6.1.4 Use case 2: Retrieve Product Category by Identifier

6.1.4.1 Interaction flow

The flow of this use case is very simple and is described in Figure 10.

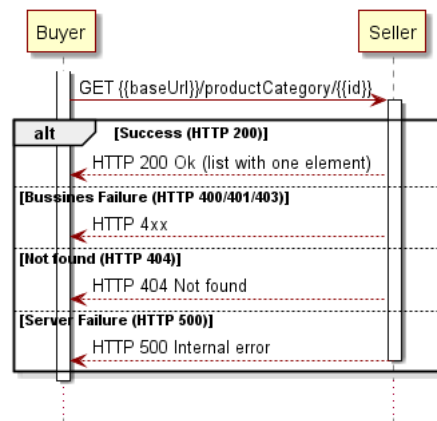


Figure 10: Use Case 2 - Retrieve Product Category by Identifier

The Buyer wants to retrieve detailed information about a single Product Category with a given `id`.

6.1.4.2. Retrieve Product Category by Identifier - Request

[R19] The Buyer must provide the `id` of the Product Category that originates from the Seller.
[MEF127 R27]

```
https://mef.com:8080/mefApi/sonata/productCatalog/v1/category/productCategory-2
```

The example above shows a Buyer's request to get the Product Category with `id` equal to `productCategory-2`. The correct response (HTTP code `200`) in the response body contains a single `ProductCategory` object matching the given `id`.

6.1.4.3. Retrieve Product Category by Identifier - Response

The snippet below presents an example of the Retrieve Product Category Request:

Retrieve `ProductCategory` by Identifier Response

```
{
  "id": "productCategory-2",
  "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productCategory/productCategory-2",
  "name": "Access E-Lines Product Offerings",
  "description": "This category groups are available Access E-Line Product Offerings",
  "lastUpdate": "2023-01-19T16:33:20.324Z",
  "lifecycleStatus": "active",
}
```



```

    "parentCategory":
    {
        "id": "productCategory-1",
        "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/category/productCategory-1"
    },
    "productOffering":
    [
        {
            "id": "productOffering-1",
            "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-1"
        },
        {
            "id": "productOffering-2",
            "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-2"
        }
    ]
}

```

[R20] The Seller **MUST** put the following attributes into the `ProductCategory` object in the response: [MEF127 R28, MEF127 R30]:

- `id`
- `name`
- `description`
- `lastUpdate`
- `lifecycleStatus`

[R21] The Seller response **MUST** include the following attributes into the `ProductCategory` if they are set by the Seller: [MEF127 R29, MEF127 R30]:

- `parentCategory`
- `subCategory`
- `productOffering`

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

6.2. Product Offering Use Cases

6.2.1 Product Offering - Model

6.2.1.1 Introduction to the model

Figure 11 presents the data model of the Product Offering. The model of the retrieve list response (`ProductOffering_Find`) is a subset of the `ProductOffering` model and contains only those attributes that can (or must) be returned by the Seller. For visibility of these differences, the `ProductOffering_Common` has been introduced. Though, it is not to be used directly in the response to any endpoint

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

- `parentCategory` category reference may be changed,
- `channel` only adding new channels is allowed,
- `agreement` may be changed to any other text,
- `marketSegment` only adding new market segments is allowed,
- `region` only adding new regions is allowed,
- `category` adding and removing of categories is allowed,
- `statusTransitions` only adding new records is allowed,
- `productOfferingStatusReason` may be updated to other text, but only when the `lifecycleStatus` is being changed
- `attachment` only adding new attachments is allowed,
- `relatedContactInformation` all types of operations are allowed,
- `productOfferingTerm` only adding new terms is allowed,
- `note` only adding new notes is allowed.

If the Product Offering hasn't reached `active` status, which implies that is still in the testing phase, then any kind of changes are allowed regarding the list defined in [O5].

[CR7]<[O5] The Buyer **MUST** update the `productOfferingStatusReason` attribute whenever the `lifecycleStatus` attribute is changed.

[R22] The Seller **MUST NOT** update the following Product Offering attributes in any of the Product Offering statuses [MEF127 R34]:

- `id`
- `productSpecification`

Product Offerings are designed to be used for exposing the particular Product Specification to the market, therefore Product Specification related attributes must not change.

[R23] Because attributes defined in [R22] are not allowed to be changed, the Seller **MUST** create a new Product Offering (with new `id`) to introduce a new version of the considered Product Offering that modifies those attributes. [MEF127 R32]

It's at the Seller's discretion to transition the older version of Product Offering to the state that prevents using that offering for ordering.

[R24] The following attributes of `statusTransitions` entities **MUST** be set [MEF127 R35]:

- `transitionDate`
- `transitionLifecycleStatus`

[D1] Whenever the `transitionDate` has passed but the transition has not taken place, then the Seller **SHOULD** update the `transitionDate` with the new date of transition.

[R25] When the planned transition took place, then the Seller **MUST NOT** remove the corresponding record from the `statusTransitions` list.

The `statusTransitions` records are used for historical log purposes.

[R26] The following attributes of `region` entities **MUST** be set [MEF127 R36]:

- `country`

[R27] The following attributes of `productOfferingTerm` entities **MUST** be set [MEF127 R37]:

- `name`
- `duration`
- `endOfTermAction`

[R28] If the `endOfTermAction` attributes of `productOfferingTerm` is set to `roll` then attribute `rollInternal` **MUST** be set. [MEF127 R38]

6.2.1.2 Product Offering Specification Schema

The concept of subschema was introduced to allow defining the Product Specification schema in the context of a particular Product Offering which is expressed as the `productOfferingSpecification` attribute. The `JSON subschema` term is introduced in [Chapter 2] of this document.

Recall, that by the definition all attributes in the Product Specification schema are optional, some of them are enumerated, some of them should satisfy the given regular expression, etc. For Product Offering purposes some attributes may be restricted to be mandatory (especially for particular business functions and product actions), the set of enumerated values may be constrained, etc. Subschema was introduced to express such constraints in an unambiguous form.

The below table defines the list of changes that could be applied to the schema and the rules for expressing them:

The change on attribute	How to express in schema	Reference requirement	Remarks
Make required	Add attribute name to <code>required</code> array.	[MEF127 R5]	
Make not applicable	Remove from the schema.	--not covered--	
Fix the value	Set fixed value as <code>const</code> .	[MEF127 R10]	
Fix the already enumerated value	Remove <code>enum</code> array, set fixed value as <code>const</code>	[MEF127 R10]	
Apply enumeration	Set enumerated values as <code>enum</code> array.	--not covered--	

The change on attribute	How to express in schema	Reference requirement	Remarks
Narrow existing enumeration	Set narrowed enumerations as <code>enum</code> array.	--not covered--	
Set default value	Set default value as <code>default</code> .	[MEF127 R3], [MEF127 R8]	If it's not set already.

Table 7. Schema modifications rules

[R29] The Seller **MUST** respect the rules defined in Table 7 for the `productOfferingSpecification` schema. [MEF127 R1], [MEF127 R2], [MEF127 R3]

[R30] The Seller **MUST** apply the agreed default value for an Optional Product-Specific Attribute if a value is not included by the Buyer in the corresponding API request. [MEF127 R8]

Let's consider the JSON Schema of Product Specification `AccessElineOvc` as the schema that will be used to construct `productOfferingSpecification` JSON Subschema (the below schema is just part of the `AccessElineOvc` schema for readability purposes):

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "allOf": [
    {
      "$ref": "#/definitions/AccessElineOvcCommon"
    },
    {
      "type": "object",
      "properties": {
        "uniEp": {
          "$ref": "#/definitions/AccessElineOvcEndPoint",
          "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the UNI. The UNI OVC End Point must be included in the Access E-Line Product."
        },
        "enniEp": {
          "$ref": "#/definitions/AccessElineOvcEndPoint",
          "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the ENNI. The ENNI OVC End Point must be included in the Access E-Line Product."
        }
      },
      "required": [
        "enniEp",
        "uniEp"
      ]
    }
  ],
  "definitions": {
    "AccessElineOvcCommon": {
      "type": "object",
      "description": "..",
      "properties": {
        "maximumFrameSize": {
          "type": "integer",
          "minimum": 1526,
          "description": "..."
        },
        "ceVlanIdPreservation": {
          "type": "string",
          "enum": [
            "PRESERVE",
            "STRIP",
            "RETAIN"
          ]
        }
      }
    }
  }
}
```

```

        "description": "...",
    },
    "cTagPcpPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "...",
    },
    "cTagDeiPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "...",
    },
    "listOfClassOfServiceNames": {
        "type": "array",
        "items": {
            "type": "string"
        },
        "minItems": 1,
        "uniqueItems": true,
        "description": "...",
    },
    "carrierEthernetSls": {
        "type": "array",
        "items": {
            "$ref": "#/definitions/CarrierEthernetSls"
        },
        "maxItems": 1,
        "minItems": 0,
        "uniqueItems": true,
        "description": "...",
    },
    "frameDisposition": {
        "$ref": "#/definitions/FrameDisposition",
        "description": "...",
    },
    "availableMegLevel": {
        "$ref": "#/definitions/AvailableMegList",
        "description": "...",
    },
    "ovcL2cpAddressSet": {
        "$ref": "#/definitions/L2cpAddressSet",
        "description": "...",
    }
}
},
... // the rest of the schema
}
}

```

For example, the task is to create a Product Offering of the `AccessElineOvc` with the following assumptions:

- class of service is constrained to the only value `Excellence`,
- attribute `maximumFrameSize` is fixed to `9100`,
- attribute `ceVlanIdPreservation` is forbidden to be used,
- attributes `cTagPcpPreservation`, `frameDisposition` are mandatory

The assumptions above may be expressed as the JSON Subschema of the Product Specification `AccessElineOvc` i.e. `productOfferingSpecification` JSON Schema:

```

{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "allOf": [
    {
      "$ref": "#/definitions/AccessElineOvcCommon"
    },
    {
      "type": "object",
      "properties": {
        "uniEp": {
          "$ref": "#/definitions/AccessElineOvcEndPoint",
          "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC at the UNI. The UNI

```

```

OVC End Point must be included in the Access E-Line Product."
    },
    "enniEp": {
      "$ref": "#/definitions/AccessElineOvcEndPoint",
      "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the ENNI. The
ENNI OVC End Point must be included in the Access E-Line Product."
    }
  },
  "required": [
    "enniEp",
    "uniEp"
  ]
}
],
"definitions": {
  "AccessElineOvcCommon": {
    "type": "object",
    "description": "..",
    "properties": {
      "maximumFrameSize": {
        "type": "integer",
        "minimum": 1526,
        "const": 9100,
        "description": "..."
      },
      "cTagPcpPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "..."
      },
      "cTagDeiPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "..."
      },
      "listOfClassOfServiceNames": {
        "type": "array",
        "items": {
          "type": "string"
        },
        "minItems": 1,
        "uniqueItems": true,
        "description": "...",
        "const": "Excellence"
      },
      "carrierEthernetSls": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/CarrierEthernetSls"
        },
        "maxItems": 1,
        "minItems": 0,
        "uniqueItems": true,
        "description": "..."
      },
      "frameDisposition": {
        "$ref": "#/definitions/FrameDisposition",
        "description": "..."
      },
      "availableMegLevel": {
        "$ref": "#/definitions/AvailableMegList",
        "description": "..."
      },
      "ovcL2cpAddressSet": {
        "$ref": "#/definitions/L2cpAddressSet",
        "description": "..."
      }
    },
    "required": ["cTagPcpPreservation", "frameDisposition"]
  },
  ... // the rest of the schema
}
}

```

[R31] Attribute `productOfferingSpecification` **MUST** be the subschema of the `sourceSchema` of `ProductSpecification`.

6.2.1.3 Product Offering Contextual Info

Product Offering Contextual Info was introduced to express how the `productOfferingSpecification` JSON Schema should be applied to particular Business Functions and Product Actions in the meaning of attributes maturity.

It's driven by the business need to not provide attributes that are redundant for particular Business Functions and Product Actions.

For better understanding, the below examples show potential use cases:

- attribute is not required during the Product Order Qualification, because it does not impact the result. At the same time, this attribute is required by the Product Ordering function,
- attribute must not be changed during the modification of the existing product.

Product Offering Contextual Info is expressed as a pair:

- `contextSchema` which is a JSON Schema,
- `context` which is a `businessFunction` and `productAction` pair.

Notice that `contextSchema` is JSON Subschema of `sourceSchema` just as `productOfferingSpecification` schema.

[R32] The following attributes of `productOfferingContextualInfo` entities **MUST** be set [MEF127 R39]:

- `businessFunction`
- `productAction`
- `productOfferingContextualSchema`

[R33] For each `productOfferingContextualInfo`, attribute `contextSchema` **MUST** be the subschema of the `sourceSchema` of the corresponding Product Specification.

[R34] For each `productOfferingContextualInfo`, attribute `contextSchema` **MUST** be the subschema of the `productOfferingSpecification` of the corresponding Product Offering.

[R35] The Seller **MUST** respect the rules defined in Table 7 for each `contextSchema` schema.

Continuing the example from the [Chapter 6.1.1.2], the task is to express that for Product Offering Qualification:

- attribute `cTagDeiPreservation` becomes required,
- attributes `availableMegLevel`, `ovcl2cpAddressSet`, `carrierEthernetSls`, `cTagDeiPreservation`, `cTagPcpPreservation` are not applicable and
- for other cases `productOfferingSpecification` JSON Schema should be applied.

The snippet below presents the above use case.

For the clarity of the example the embedded schemas (i.e. `contextSchema` attributes) were replaced with references that are placed below this example. Additionally, the embedded schemas were unescaped.

```
{
  ...
  "productOfferingContextualInfo":
  [
    {
      "contextSchema": {
        // "#context-schema-1"
      },
      "context":
      {
        "productAction": "all",
        "businessFunction": "all"
      }
    },
    {
      "contextSchema": {
        // "#context-schema-2"
      },
      "context":
      {
        "productAction": "all",
        "businessFunction": "productOfferingQualification"
      }
    }
  ]
  ...
}
```

The above example presents the list of the `productOfferingContextualInfo` that express that `#context-schema-1` should be used for any combination of `businessFunction` and `productAction` except the case where `businessFunction=productOfferingQualification` then `#context-schema-2` should be used instead.

Embedded schemas:

- `#context-schema-1`

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "allOf": [
    {
      "$ref": "#/definitions/AccessElineOvcCommon"
    },
    {
      "type": "object",
      "properties": {
        "uniEp": {
          "$ref": "#/definitions/AccessElineOvcEndPoint",
          "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the UNI. The UNI OVC End Point must be included in the Access E-Line Product."
        },
        "enniEp": {
          "$ref": "#/definitions/AccessElineOvcEndPoint",
          "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the ENNI. The ENNI OVC End Point must be included in the Access E-Line Product."
        }
      },
      "required": [
        "enniEp",
        "uniEp"
      ]
    }
  ],
  "definitions": {
    "AccessElineOvcCommon": {
```

```

    "type": "object",
    "description": "..",
    "properties": {
      "maximumFrameSize": {
        "type": "integer",
        "minimum": 1526,
        "const": 9100,
        "description": "..."
      },
      "cTagPcpPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "..."
      },
      "cTagDeiPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "..."
      },
      "listOfClassOfServiceNames": {
        "type": "array",
        "items": {
          "type": "string"
        },
        "minItems": 1,
        "uniqueItems": true,
        "description": "...",
        "const": "Excellence"
      },
      "carrierEthernetSls": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/CarrierEthernetSls"
        },
        "maxItems": 1,
        "minItems": 0,
        "uniqueItems": true,
        "description": "..."
      },
      "frameDisposition": {
        "$ref": "#/definitions/FrameDisposition",
        "description": "..."
      },
      "availableMegLevel": {
        "$ref": "#/definitions/AvailableMegList",
        "description": "..."
      },
      "ovcL2cpAddressSet": {
        "$ref": "#/definitions/L2cpAddressSet",
        "description": "..."
      }
    },
    "required": ["cTagPcpPreservation", "frameDisposition"]
  },
  ... // the rest of the schema
}

```

Schema `#context-schema-1` is equal to the `productOfferingSpecification` schema. That means that for every use case (except `businessFunction=productOfferingQualification`) schema is unchanged.

- `#context-schema-2`

```

{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "allOf": [
    {
      "$ref": "#/definitions/AccessElineOvcCommon"
    },
    {
      "type": "object",
      "properties": {
        "uniEp": {
          "$ref": "#/definitions/AccessElineOvcEndPoint",
          "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the UNI. The UNI OVC End Point must be included in the Access E-Line Product."
        }
      }
    }
  ]
}

```

```

    },
    "enniEp": {
      "$ref": "#/definitions/AccessElineOvcEndPoint",
      "description": "MEF 26.2 sec. 16 - The OVC EP object for the OVC EP at the ENNI. The ENNI OVC End Point must be included in the Access E-Line Product."
    }
  },
  "required": [
    "enniEp",
    "uniEp"
  ]
}
],
"definitions": {
  "AccessElineOvcCommon": {
    "type": "object",
    "description": "..",
    "properties": {
      "maximumFrameSize": {
        "type": "integer",
        "minimum": 1526,
        "const": 9100,
        "description": "..."
      },
      "listOfClassOfServiceNames": {
        "type": "array",
        "items": {
          "type": "string"
        },
        "minItems": 1,
        "uniqueItems": true,
        "description": "...",
        "const": "Excellence"
      },
      "frameDisposition": {
        "$ref": "#/definitions/FrameDisposition",
        "description": "..."
      },
      "cTagDeiPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "..."
      },
      "cTagPcpPreservation": {
        "$ref": "#/definitions/EnabledDisabled",
        "description": "..."
      }
    }
  },
  "required": ["cTagPcpPreservation", "frameDisposition", "cTagDeiPreservation"]
},
... // the rest of the schema
}
}

```

The above example is the realization of the [O7].

Schema `#context-schema-2` is modified according to the example described above. Below list summarizes the modifications applied to the `productOfferingSpecification` schema that results with `#context-schema-2`:

- attribute `cTagDeiPreservation` is required, therefore is added to the `required` array,
- attributes `availableMegLevel`, `ovcl2cpAddressSet`, `carrierEthernetSls`, `cTagPcpPreservation` are not applicable, therefore were removed from the schema

[R36] The Seller **MUST** return `productOfferingContextualInfo` list that covers all possible variants of Business Functions and Product Actions. [MEF127 R33]

It doesn't mean that the cartesian product of possibilities must be returned. Wildcards are recommended to be used to cover the whole group of possibilities, as is done in the example

above.

[O7] The Seller **MAY** use the `productAction=all`, `businessAction=all` wildcards to cover all use cases and extend the `productOfferingContextualInfo` by adding specific use cases.

[R37] If the Buyer sends the request for any combination of `businessFunction` and `productAction`, the request **MUST** be valid against the schema defined in the corresponding `productOfferingContextualInfo` attribute of referred `ProductOffering`.

[R38] The Buyer and the Seller **MUST** agree on whether the Buyer can include in an API request Product-Specific Attributes that have been classified as Fixed. [MEF127 R10]

[R39] If the Buyer and Seller agree that Product-Specific Attributes classified as Fixed cannot be included in API request, the Buyer and Seller **MUST** agree on whether the Seller includes Product-Specific Attributes classified as Fixed in the corresponding API responses. [MEF127 R11]

[R40] If the Buyer and Seller agree that Product-Specific Attributes classified as Fixed cannot be included in an API request, the Seller **MUST** reject an API request from the Buyer if it includes a Product-Specific Attribute that has been classified as Fixed for the Business Function (POQ, Quote, Order), Product Action (add, modify), and Product Offering. [MEF127 R12]

[R41] If the Buyer and Seller agree that Product-Specific Attributes classified as Fixed cannot be included in an API request, and if a Product-Specific Attribute is classified to be Fixed for Inventory for a Product Offering, then the Seller **MUST NOT** include a value for the attribute in the corresponding API response. [MEF127 R13]

[R42] If the Buyer and Seller agree that Product-Specific Attributes classified as Fixed can be included in an API request, the Seller **MUST** reject an API request from the Buyer if it includes a Product-Specific Attribute that has been classified as Fixed for the Business Function (POQ, Quote, Order), Product Action (add, modify), and Product Offering and includes a value that is different than the agreed-on fixed value. [MEF127 R14]

[R43] If the Buyer and Seller agree that Product-Specific Attributes classified as Fixed can be included in API request, and if a Product-Specific Attribute is agreed to be Fixed for Inventory for a Product Offering, then the Seller **MUST** include a value for the Product-Specific attribute in the Inventory API response. [MEF127 R15]

6.2.2 Product Offering - Lifecycle

Figure 12 presents the Product Offering state machine:

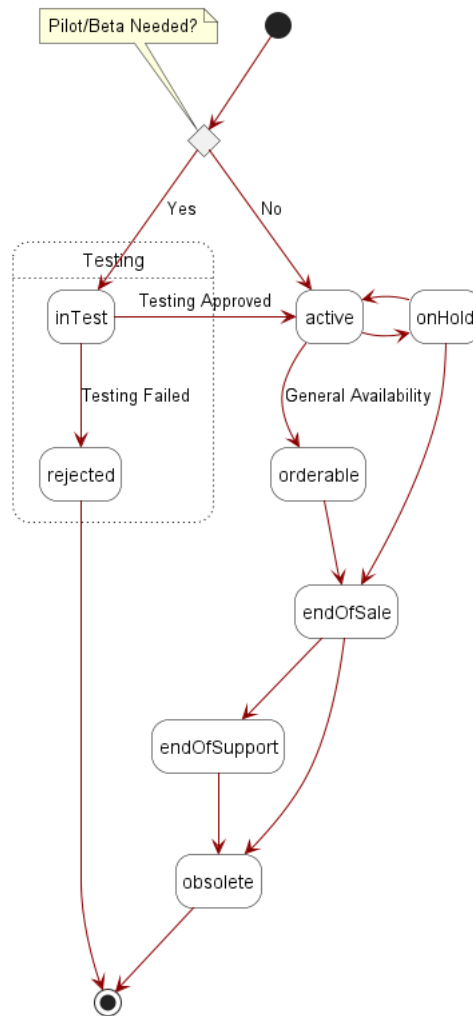


Figure 12. Product Offering State Machine

The Product Offering State Machine is simpler than the one proposed by TMF [TMF620] because it focuses on exposing Product Offerings to the Buyers, not the whole Product Offering Management. The specific states and notifications are managed by the Seller.

[O8] The Seller **MAY** decide for agreed Buyers being part of beta test process to expose the Product Offerings in `inTest` state.

Table 6 presents the mapping between API `lifecycleStatus` values (aligned with TMF) and MEF127 naming together with states' descriptions.

<code>lifecycleStatus</code>	MEF 127 name	Description
<code>active</code>	ACTIVE	When a Product Offering has been defined and will be made available for ordering; however, it is not yet generally available.
<code>endOfSale</code>	END_OF_SALE	The <code>endOfSale</code> status means the Product Offering cannot be Installed by any new or existing Buyers, but Buyers may still have Products in use and may Change or Disconnect them, and receive support.

lifecycleStatus	MEF 127 name	Description
endOfSupport	END_OF_SUPPORT	When a Product Offering in the endOfSale status is no longer supported, the status transitions to endOfSupport . Any existing products can no longer be Changed, with the only Order action allowed is Disconnect.
obsolete	OBSOLETE	After a Product Offering that is no longer available it transitions to obsolete and may be removed at the Seller's discretion from the Product Catalog . This is a final state.
onHold	ON_HOLD	A Product Offering that has been orderable , but is currently not available for Buyers due to supply constraints, product recall, or other issues preventing it to be offered.
orderable	ORDERABLE	A new Product Offering is in the orderable state when it is available for ordering by Buyers.
inTest	PILOT_BETA	When a Product Offering starts Pilot/Beta testing, it starts in the inTest state.
rejected	REJECTED	When Pilot/Beta testing fails the Product Offering or Product Specification transitions to the rejected state. This is a final state.

Table 8. Product Offering lifecycle statuses

[R44] The Seller **MUST** support all Statuses for Product Offering and the associated state transitions [MEF127 R66].

It is one the Seller discretion whether the **inTest** and **rejected** state will be used.

[O9] The Seller **MAY** update the Product Offering (see chapter [6.2.1] to find updatable attributes) when the Product Offering **lifecycleStatus** is not equal to **endOfSale**, **endOfSupport**, **obsolete**.

[R45] The Seller **MUST NOT** update the Product Offering when it is in the final state **obsolete**.

[O10] The Seller **MAY** remove a Product Offering that is in the **obsolete** state from the Product Catalog. [MEF127 O9]

Removing obsoleted Product Offerings is on the Seller's discretion. Nevertheless it must be consulted with the Buyer for the reasons when the Buyer still uses their definitions for historical purposes.

[O11] The Seller **MAY** remove a Product Offering that is in the `rejected` state from the Product Catalog. [MEF O10]

6.2.3 Use case 3: Retrieve Product Offering List

6.2.3.1 Interaction flow

The flow of this use case is very simple and is described in Figure 13.

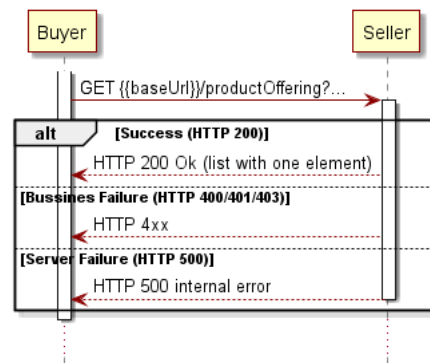


Figure 13: Use Case 3 - Retrieve Product Offering List

6.2.3.2. Retrieve Product Offering List - Request

[O12] The Buyer **MAY** retrieve the list of Product Categories by using a `GET /productOffering` operation with desired filtering criteria. The attributes that are available to be used are: [MEF127 O6]:

- `name`
- `lastUpdate.gt`
- `lastUpdate.lt`
- `lifecycleStatus`
- `agreement`
- `channel`
- `marketSegment`
- `region.country`
- `category.id`

[CR8]<[O12] The Seller **MUST** reject the request if the attributes requested by the Buyer are not supported value defined in [O12]. [MEF127 R4]

The Buyer may also ask for pagination with the use of the `offset` and `limit` parameters. The filtering and pagination attributes must be specified in URI query format [RFC3986](#). Section [7.1.2](#). provides details about the implementation of pagination mechanism.

Attributes `marketSegment` and `channel` are collections of primitive types (in this particular case `List<String>`). For implementation purposes, the strategy of querying interpretation is settled as:

- Requesting Entity may provide any number of desired values in filtering criteria provided as query parameters ([RFC3986](#) with the standard limitation of URIs length) respecting the following notation:

```
?marketSegment=Wholesale&marketSegment=Federal&...
```

- Provided values are interpreted as the alternatives,
- Resource is matched if any of provided values match any value from the collection.

[R46] If the Buyer provides more than one value in filtering criteria for the array type attributes, then the Seller **MUST** return all Product Offerings whose corresponding attributes contain any of provided values.

Regarding [R46], if the Buyer provides e.g. `marketSegment=Federal&marketSegment=Financial` then the Seller uses that values as alternatives i.e. result matches if at least one of the provided values is contained by the Product Offering's `marketSegment` list.

```
http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering?
status=orderable&marketSegment=Federal&marketSegment=Financial&limit=10&offset=0
```

The example above shows a Buyer's request to get the first ten Product Offerings that are in `orderable` status and are available for the `Federal` or `Financial` markets. The correct response (HTTP code `200`) in the response body contains a list of `ProductOffering_Find` objects matching the criteria. To get more details (e.g. the list of 'Product Offering Terms'), the Buyer has to query a specific `ProductOffering` by `id`.

6.2.3.3. Retrieve Product Offering List - Response

The snippet below presents an example of the Retrieve Product Offering Lis response:

Retrieve `ProductOffering` List Response

Headers:

```
X-Result-Count=1
X-Total-Count=1
X-Pagination-Throttled=false
```

Body:

```
[
  {
    "id": "productOffering-1",
    "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-1",
    "name": "Access E-line OVC Basic",
    "lastUpdate": "2023-01-19T16:33:20.324Z",
    "lifecycleStatus": "orderable",
```



```

    "agreement": "Official agreement no 4",
    "channel": [
      "DirectSales",
      "Distribution"
    ],
    "marketSegment": [
      "Federal",
      "Financial"
    ],
    "region": [
      {
        "stateOrProvince": "Malopolskie",
        "country": "Poland"
      }
    ],
    "category": [
      {
        "id": "productCategory-2",
        "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/category/productCategory-2"
      }
    ]
  },
  {
    "id": "productOffering-2",
    "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-2",
    "name": "Access E-line OVC Excellence",
    "lastUpdate": "2023-01-19T16:33:20.324Z",
    "lifecycleStatus": "orderable",
    "agreement": "string",
    "channel": [
      "DirectSales"
    ],
    "marketSegment": [
      "Federal"
    ],
    "region": [
      {
        "stateOrProvince": "Malopolskie",
        "country": "Poland"
      }
    ],
    "category": [
      {
        "id": "productCategory-2",
        "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/category/productCategory-2"
      }
    ]
  }
]

```

[R47] When the Buyer queries by `category.id` attribute, then the Seller **MUST** include every Product Offering that is the direct member and indirect member of this category. [MEF127 R44]

By indirect member of the Product Category should be understood the member that its category is in `subcategories` list of that category. This rule applies recursively.

Assume that Category C is the subcategory of Category B and Category B is the subcategory of Category A. Then the Product Offering that is categorized by Category C is:

- a direct member of Category C,
- an indirect member of Category B (because Category C is a subcategory of Category B),
- an indirect member of Category A (because Category C is a subcategory of Category A by recursiveness).

[R48] The Seller **MUST** put the following attributes into the `ProductOffering_Find` object in the response: [MEF127 R40, MEF127 R45]:

- `id`
- `name`
- `lastUpdate`
- `lifecycleStatus`
- `agreement`
- `channel`
- `marketSegment`
- `region`
- `category`

[R49] The Seller response **MUST** include every Product Offering where the `channel` filter criteria match one of the Product Offering's `channel` or the Product Offering's `channel` is an empty list [MEF127 R41]:

[R50] The Seller response **MUST** include every Product Offering where the `marketSegment` filter criteria match one of the Product Offering's `marketSegment` or the Product Offering's `marketSegment` is an empty list [MEF127 R42].

[R51] The Seller response **MUST** include every Product Offering where the `region.country` filter criteria match one of the Product Offering's `region.country` or the Product Offering's `region.country` is an empty list [MEF127 R43].

[R52] If case no items matching the criteria are found, the Seller **MUST** return a valid (HTTP code `200`) response with an empty list. [MEF127 R46]

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

Note: The Product Offering model for this use case is the subset of the model from the chapter [\[6.2.1\]](#).

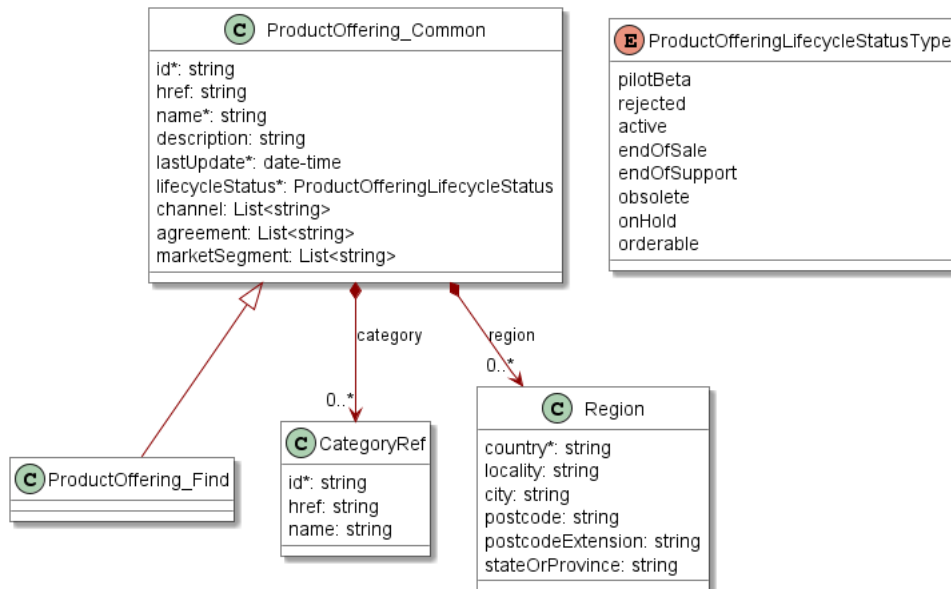


Figure 14: Use Case 3 - Product Offering Find Model

6.2.4 Use case 4: Retrieve Product Offering by Identifier

6.2.4.1 Interaction flow

The flow of this use case is very simple and is described in Figure 14.

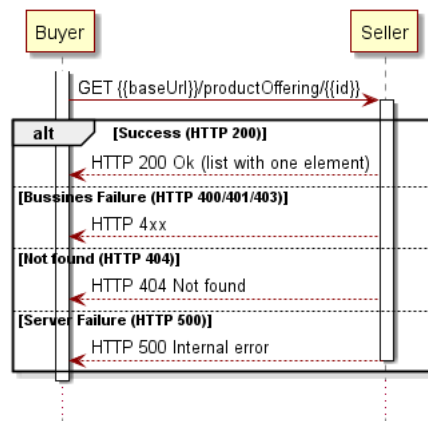


Figure 15: Use Case 4 - Retrieve Product Offering by Identifier

The Buyer wants to retrieve detailed information about a single Product Offering with the given `id`.

6.2.4.2. Retrieve Product Offering by Identifier - Request

[R53] The Buyer must provide the `id` of the Product Offering that originates from the Seller.
[MEF127 R47]

<http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-1>

The example above shows a Buyer's request to get the Product Category with `id` equal to `productOffering-1`. The correct response (HTTP code `200`) in the response body contains a single `ProductOffering` object matching the given `id`.

6.2.4.3. Retrieve Product Offering by Identifier - Response

The snippet below presents an example of the Retrieve Product Offering Request (the schemas were not provided intentionally due to the example's clarity reasons):

Retrieve `ProductOffering` by Identifier Response

```
{
  "id": "productOffering-1",
  "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productOffering/productOffering-1",
  "name": "Access E-line OVC Basic",
  "lastUpdate": "2023-01-19T16:33:20.324Z",
  "lifecycleStatus": "orderable",
  "agreement": "Official agreement no 4",
  "channel": [
    "DirectSales",
    "Distribution"
  ],
  "marketSegment": [
    "Federal",
    "Financial"
  ],
  "region": [
    {
      "stateOrProvince": "Malopolskie",
      "country": "Poland"
    }
  ],
  "category": [
    {
      "id": "productCategory-2",
      "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/category/productCategory-2"
    }
  ],
  "statusTransitions": [
    {
      "transitionDate": "2024-01-19T16:33:20.324Z",
      "transitionLifecycleStatus": "obsolete"
    }
  ],
  "productOfferingStatusReason": "Ready to be used for ordering.",
  "attachment": [
    {
      "attachmentId": "4e5b3701-47f8-47a7-bcf8-d3d740b4cd60",
      "author": "John Doe",
      "creationDate": "2023-01-19T16:33:20.324Z",
      "description": "Signed contract",
      "mimeType": "application/pdf",
      "name": "Customers contract",
      "size": {
        "amount": 105.0,
        "units": "KBYTES"
      },
      "source": "buyer",
      "url": "http://some-domain.com/attachment/file/4e5b3701-47f8-47a7-bcf8-d3d740b4cd60"
    }
  ],
  "productOfferingTerm": [
    {
      "description": "Basic Term",
      "duration": {
        "amount": 12,
        "units": "calendarMonths"
      },
      "endOfTermAction": "roll",
      "name": "Basic",
      "rollInterval": {
```

```

        "amount": 6,
        "units": "calendarMonths"
    }
},
"note": [
    {
        "author": "John Doe",
        "date": "2023-01-19T16:33:20.324Z",
        "id": "43072c06-34ac-4713-b0b1-371cb7479400",
        "source": "buyer",
        "text": "Lorem ipsum"
    }
],
"productSpecification": {
    "id": "productSpecification-1",
    "href":
"http://mef.com:8080/mefApi/sonata/productCatalog/v1/productSpecification/productSpecification-1"
},
"productOfferingContextualInfo":
[
    {
        "contextSchema": {
            // here should be the schema
        }
        "context":
        {
            "productAction": "all",
            "businessFunction": "all"
        }
    }
]
}

```

[R54] The Seller **MUST** put the following attributes into the `ProductOffering` object in the response: [MEF127 R48, MEF127 R50]:

- `id`
- `name`
- `description`
- `lastUpdate`
- `lifecycleStatus`
- `productOfferingSpecification`
- `productSpecification`

[R55] The Seller response **MUST** include the following attributes in the `ProductOffering` if they are set by the Seller: [MEF127 R49, MEF127 R50]:

- `statusTransitions`
- `productOfferingStatusReason`
- `agreement`
- `attachment`
- `relatedContactInformation`
- `channel`
- `marketSegment`
- `region`
- `productOfferingTerm`
- `note`
- `category`

- `productOfferingContextualInfo`

[R56] The Seller response **MUST** include exactly one of the `productOfferingSpecification` attributes:

- `schema`
- `schemaLocation`

[R57] The Seller response **MUST** include exactly one of the `contextSchema` attributes:

- `schema`
- `schemaLocation`

Attributes `schema` and `schemaLocation` are modeled as mutually exclusive to avoid the dualistic representation of `productOfferingSpecification` or `contextSchema`.

[R58] If the `attachment` is provided, either the `attachment.url` or (`attachment.content` and `attachment.mimeType`) **MUST** be specified.

[R59] For Product Offerings, the Seller **MUST** set the respective `source=seller` attribute when adding any item to one of the following lists: `note`, `attachment`.

The source of the notes is always the Seller (`note[?].source=seller`) because API doesn't allow for any modifications that could be initiated by the Buyer.

Note: The Product Offering model for this use case is the full set of the model from the chapter [6.2.1]. To see the model go to the mentioned chapter.

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

6.3. Product Specification Use Cases

6.3.1 Product Specification - Model

Figure 16 presents the data model of the Product Specification. The model of the retrieve list response (`ProductSpecification_Find`) is a subset of the `ProductSpecification` model and contains only those attributes that can (or must) be returned by the Seller. For visibility of this differences the `ProductSpecification_Common` has been introduced. Though, it is not to be used directly in the response of any endpoint.

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

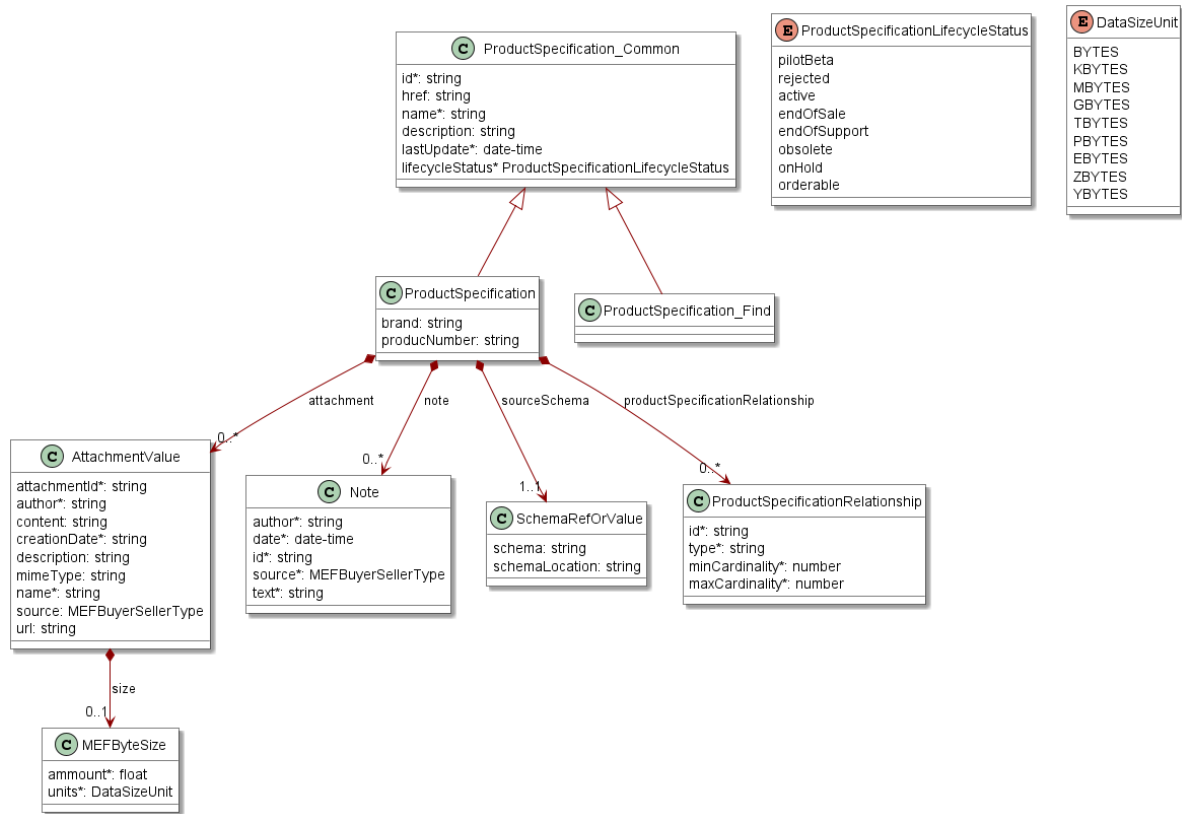


Figure 16. Product Specification Model

[O13] The Seller **MAY** update the following Product Specification attributes:

- **description**
- **lifecycleStatus**
- **attachment**
- **note**
- **productSpecificationRelationship**

[CR9]<[O8] In the case of a change of any attribute defined in [O8], the Seller **MUST** update the **lastUpdate** attribute with the date of the modification. [MEF127 R51]

[CR10]<[O8] If the Product Specification's **lifecycleStatus** reached **active** status, then the changes introduced by the Seller **MUST** respect the backward compatibility for Product Specification, which is defined as:

- **description** may be changed to any other text,
- **lifecycleStatus** may be changed accordingly to the state machine,
- **attachment** only adding new attachments is allowed,
- **note** only adding new notes is allowed,
- **productSpecificationRelationship** adding new relationships whose **minCardinality** attribute is equal to 0 is allowed only.

[R60] The Seller **MUST NOT** update the Product Specification attributes:

- **id**
- **name**

- brand
- productNumber
- sourceSchema

[R61] The following attributes of `productSpecificationRelationship` list elements **MUST** be set [MEF127 R52]:

- id
- relationshipType
- minCardinality
- maxCardinality

Product Specifications are designed to be used for decoration of the Product Schema (represented by `sourceSchema`) by the business attributes to publish them for commercial usage. Related `sourceSchema` defines the JSON schema (`JSONSchema`) of a prospective Product instance. At the very beginning, by definition all attributes in the `sourceSchema` are optional and during integration between the Buyer and Seller must be negotiated and defined in `ProductOffering` as `productOfferingSpecification` and `productOfferingContextualInfo` attributes.

6.3.2 Product Specification - Lifecycle

Figure 17 presents the Product Specification state machine:

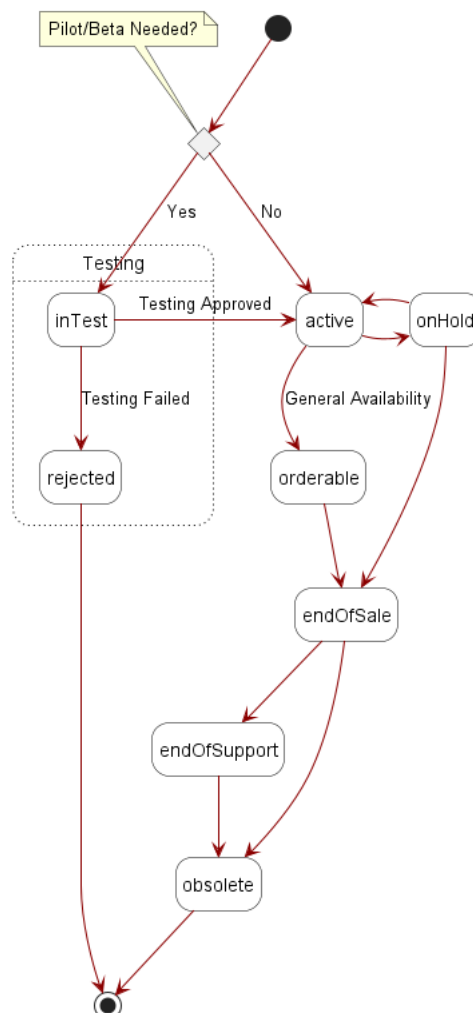


Figure 17. Product Offering State Machine

The Product Specification State Machine is consistent with the State Machine of Product Offering and it is a simplification of the one proposed by TMF [TMF620] because it focuses on exposing Product Specification to the Buyers, not the whole Product Specifications Management. The specific states and notifications are managed by the Seller.

[O14] The Seller **MAY** decide for agreed Buyers being part of beta test process to expose the Product Specifications in **inTest** state.

Table 6 presents the mapping between API **lifecycleStatus** values (aligned with TMF) and MEF127 naming together with states' description.

lifecycleStatus	MEF 127 name	Description
active	ACTIVE	When a Product Specification has been defined and will be made available for ordering; however, it is not yet generally available.
endOfSale	END_OF_SALE	The endOfSale status means that Product Specification cannot be used for the creation of new Product Offerings. Existing Product Offerings related to a given Product Specification must change their status to endOfSale too (to read more about Product Offering endOfSale status go to chapter [6.2.2] of this document)
endOfSupport	END_OF_SUPPORT	When a Product Specification in the endOfSale status is no longer supported, the status transitions to endOfSupport . Any existing products can no longer be Changed, the only Order action allowed is Disconnect.
obsolete	OBSOLETE	After a Product Specification that is no longer available, it transitions to obsolete and may be removed at the Seller's discretion from the Product Catalog. This is a final state.
onHold	ON_HOLD	A Product Specification that has been orderable , but is currently not available for Buyers due to supply constraints, product recall, or other issues preventing it to be offered.
orderable	ORDERABLE	A new Product Specification is in the orderable state when it is available for ordering by Buyers.
inTest	PILOT_BETA	When a Product Specification starts Pilot/Beta testing, it starts in the inTest state.

lifecycleStatus	MEF 127 name	Description
rejected	REJECTED	When Pilot/Beta testing fails the Product Specification to the rejected state. This is a final state.

Table 9. Product Specification lifecycle statuses

[R62] The Seller **MUST** support all Statuses for Product Specification and the associated state transitions [MEF127 R66].

[O15] The Seller **MAY** update the Product Specification (see chapter [6.3.1] to find updatable attributes) when the Product Specification lifecycleStatus is not final i.e. rejected or obsolete.

[R63] The Seller **MUST NOT** update the Product Specification when it is in the final state rejected or obsolete.

States are compared according to the following rules:

- because the State Machine of the Product Specification and Product Offering is the same, by equality of states we understood the lifecycleStatus with the same values,
- inTest is prior to any other state,
- active is prior to orderable, endOfSale, endOfSupport, obsolete,
- onHold is prior to orderable, endOfSale, endOfSupport, obsolete but it cannot be compared to active,
- orderable is prior to endOfSale, endOfSupport, obsolete,
- endOfSale is prior to endOfSupport, obsolete
- endOfSupport is prior to obsolete
- obsolete and rejected are final state and are ahead of any other states.

[R64] The Seller **MUST NOT** create a Product Offering related to the Product Specification that lifecycleStatus is equal to endOfSale, endOfSupport or obsolete.

[R65] Product Offering lifecycleStatus **MUST** be compliant with the lifecycleStatus of related Product Specification i.e. according to the State Machine Product Offering lifecycleStatus must be prior or equal to the Product Specification lifecycleStatus that is prior to orderable and must be ahead of Product Specification lifecycleStatus that is ahead of equal orderable.

Above requirement assures that Product Offerings cannot be transitioned to the states that allowed them to be used for marketing purposes when the related Product Specification is not ready for marketing. And at the same time assures that Product Offering that is already available for marketing can be obsolete.

[R66] If the statuses of Product Offering and related Product Specification are the same, and the Seller changes the Product Specification state, the Seller **MUST** update the Product Offering state to preserve compliance.

The same rule is not applying to the state modifications on Product Offerings i.e. Product Offering state may be changed and it doesn't imply a change of state of related Product Specification. At the same time, the requirements described above must be applied.

[R67] If the Seller moves Product Specification to **obsolete** state, then the Seller **MUST** move all related Product Offerings to **obsolete** state too.

[O16] The Seller **MAY** remove a Product Specification from the Product Catalog. that is only in the **obsolete** state. [MEF127 O9]

[CR11]<[O16] When the Seller is removing the **obsolete** Product Specifications, then the Seller **MUST** remove all related Product Offerings too.

Removing obsoleted Product Specifications is on the Seller's discretion. Nevertheless it must be consulted with the Buyer for the reasons when the Buyer still uses their definitions for historical purposes.

[O17] The Seller **MAY** remove a Product Specification from the Product Catalog, that is in the **rejected** state. [MEF O10]

6.3.3 Use case 5: Retrieve Product Specification List

6.3.3.1 Interaction flow

The flow of this use case is very simple and is described in Figure 18.

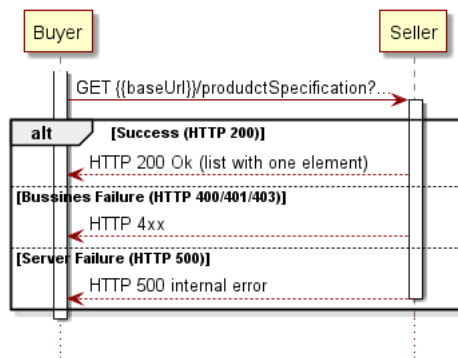


Figure 18: Use Case 5 - Retrieve Product Specification List

6.3.3.2. Retrieve Product Specification List - Request

[O18] The Buyer **MAY** retrieve the list of Product Specifications by using a **GET** `/productSpecification` operation with desired filtering criteria. The attributes that are available to be used are: [MEF127 O7]:

- `name`
- `lastUpdate.gt`
- `lastUpdate.lt`

- `lifecycleStatus`
- `brand`

[CR12]<[O18] The Seller **MUST** reject the request if the attributes requested by the Buyer are not supported value defined in [O18]. [MEF127 R4]

The Buyer may also ask for pagination with the use of the `offset` and `limit` parameters. The filtering and pagination attributes must be specified in URI query format [RFC3986](#). Section [7.1.2](#). provides details about the implementation of pagination mechanism.

```
http://mef.com:8080/mefApi/sonata/productCatalog/v1/productSpecification?
lifecycleStatus=orderable&limit=3&offset=8
```

The example above shows a Buyer's request to omit first seven and get next two Product Specifications that are in `orderable` status. The correct response (HTTP code `200`) in the response body contains a list of `ProductSpecification_Find` objects matching the criteria. To get more details (e.g. `sourceSchema`), the Buyer has to query a specific `ProductSpecification` by `id`.

6.3.3.3. Retrieve Product Specification List - Response

The snippet below presents an example of the Retrieve Product Specification List response:

Retrieve `ProductSpecification` List Response

Headers:

```
X-Result-Count=2
X-Total-Count=10
X-Pagination-Throttled=true
```

Body:

```
[
  {
    "id": "productSpecification-11",
    "href":
"http://mef.com:8080/mefApi/sonata/productCatalog/v1/productSpecification/productSpecification-11",
    "name": "Ethernet Virtual Private Tree EVC",
    "lifecycleStatus": "orderable",
    "lastUpdate": "2023-01-19T16:30:51.626Z"
  },
  {
    "id": "productSpecification-9",
    "href":
"http://mef.com:8080/mefApi/sonata/productCatalog/v1/productSpecification/productSpecification-9",
    "name": "Ethernet Private Tree EVC EP",
    "lifecycleStatus": "orderable",
    "lastUpdate": "2023-01-19T16:30:51.626Z"
  }
]
```

[R68] The Seller **MUST** put the following attributes into the `ProductSpecification_Find` object in the response: [MEF127 R53, MEF127 R54]:

- `id`
- `name`
- `lastUpdate`
- `lifecycleStatus`

[R69] If case no items matching the criteria are found, the Seller **MUST** return a valid response with an empty list. [MEF127 R55]

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

Note: The Product Specification model for this use case is the subset of the model from the chapter [\[6.3.1\]](#).

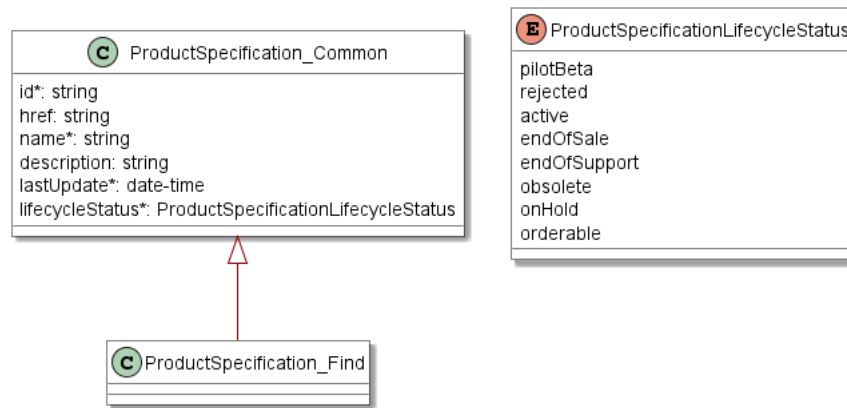


Figure 19: Use Case 5 - Product Specification Find Model

6.3.4 Use case 6: Retrieve Product Specification by Identifier

6.3.4.1 Interaction flow

The flow of this use case is very simple and is described in Figure 20.

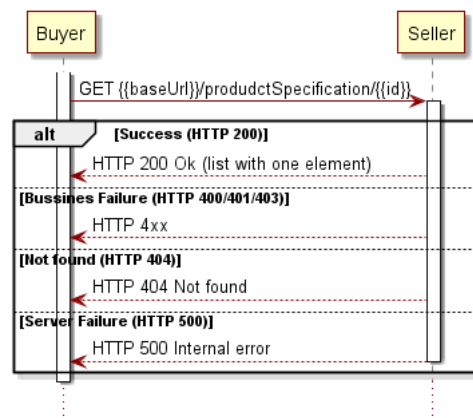


Figure 20: Use Case 6 - Retrieve Product Specification by Identifier

The Buyer wants to retrieve detailed information about a single Product Specification with a given `id`.

6.3.4.2. Retrieve Product Specification by Identifier - Request

[R70] The Buyer must provide the `id` of the Product Specification that originates from the Seller. [MEF127 R56]

```
http://127.0.0.1:8080/mefApi/sonata/productCatalog/v1/productSpecification/productSpecification-9
```

The example above shows a Buyer's request to get the Product Specification with `id` equal to `productSpecification-9`. The correct response (HTTP code `200`) in the response body contains a single `ProductSpecification` object matching the given `id`.

6.3.4.3. Retrieve Product Specification by Identifier - Response

The snippet below presents an example of the Retrieve Product Specification Request (the schemas were not provided intentionally due to the example's clarity reasons):

Retrieve `ProductSpecification` by Identifier Response

```
{
  "id": "productSpecification-9",
  "href": "http://mef.com:8080/mefApi/sonata/productCatalog/v1/productSpecification/productSpecification-9",
  "name": "Ethernet Private Tree EVC EP",
  "lifecycleStatus": "orderable",
  "lastUpdate": "2023-01-19T16:30:51.626Z",
  "description": "EP TREE EVC EP",
  "attachment": [
    {
      "attachmentId": "4e5b3701-47f8-47a7-bcf8-d3d740b4cd60",
      "author": "John Doe",
      "creationDate": "2023-01-19T16:33:20.324Z",
      "description": "EP TREE EVC doc",
      "mimeType": "application/pdf",
      "name": "Technical documentation",
      "size": {
        "amount": 1050.0,
        "units": "KBYES"
      },
      "source": "buyer",
      "url": "https://lso.mef.net/lso-payload-catalog/ep-tree-ep-product-payload"
    }
  ],
  "note": [
    {
      "author": "John Doe",
      "date": "2023-01-19T16:33:20.324Z",
      "id": "43072c06-34ac-4713-b0b1-371cb7479400",
      "source": "buyer",
      "text": "Lorem ipsum"
    }
  ],
  "sourceSchema": {
    "schemaLocation": "http://mef.com:8081/publisher/v1/schema/6346d966-0a9c-4dfe-88f2-86b0e995a8a9"
  },
  "productSpecificationRelationship": [
    {
      "id": "productSpecification-10",
      "relationshipType": "rootEndpointOfEvc",
      "minCardinality": 0,

```

```

        "maxCardinality": 1
      },
      {
        "id": "productSpecification-10",
        "relationshipType": "leafEndpointOfEvc",
        "minCardinality": 0,
        "maxCardinality": 1
      },
      {
        "id": "productSpecification-6",
        "relationshipType": "connectsToUni",
        "minCardinality": 1,
        "maxCardinality": 1
      }
    ]
  }
}

```

[R71] The Seller **MUST** put the following attributes into the `ProductSpecification` object in the response: [MEF127 R57, MEF127 R59]:

- `id`
- `name`
- `description`
- `lastUpdate`
- `lifecycleStatus`
- `sourceSchema`

[R72] The Seller response **MUST** include the following attributes in the `ProductSpecification` if they are set by the Seller: [MEF127 R58, MEF127 R59]:

- `brand`
- `productNumber`
- `attachment`
- `productSpecificationRelationship`
- `note`

The full list of attributes is available in [Section 7](#) and in the API specification which is an integral part of this standard.

[R73] If the `attachment` is provided, either the `attachment.url` or (`attachment.content` and `attachment.mimeType`) **MUST** be specified.

[R74] For Product Specifications, the Seller **MUST** set the respective `source=seller` attribute when adding any item to one of the following lists: `note`, `attachment`.

The source of the notes is always the Seller (`note[?].source=seller`) because API doesn't allow for any modifications that could be initiated by the Buyer.

Note: The Product Specification model for this use case is the full set of the model from the chapter [6.3.1]. To see the model go to the mentioned chapter.

[R75] The Seller response **MUST** include exactly one of the `sourceSchema` attributes:

- `schema`
- `schemaLocation`

Attributes `schema` and `schemaLocation` are modeled as mutually exclusive to avoid the dual nature of `sourceSchema` representation.

6.9. Use case 7: Register for Event Notifications

[O19] The Seller **MAY** support Product Catalog Event Notifications.

It's in Seller's and Buyer's discretion to decide if registration for Event Notifications is supported.

[011] The Seller **MAY** support the following Product Category related Notification Types (`ProductCategoryEventType`):

- `productCategoryCreateEvent`
- `productCategoryAttributeValueChangeEvent`
- `productCategoryAttributeValueChangeEvent`

[CR13]<[O11] If the Seller wants to support one of the `ProductCategoryEventType`, then the Seller **MUST** support of all them.

[012] The Seller **MAY** support the following Product Offering related Notification Types (`ProductOfferingEventType`):

- `productOfferingCreateEvent`
- `productOfferingAttributeValueChangeEvent`
- `productOfferingStatusChangeEvent`

[CR14]<[O12] If the Seller wants to support one of the `ProductOfferingEventType`, then the Seller **MUST** support of all them.

[013] The Seller **MAY** support the following Product Specification related Notification Types (`ProductSpecificationEventType`):

- `productSpecificationCreateEvent`
- `productSpecificationAttributeValueChangeEvent`
- `productSpecificationStatusChangeEvent`

[CR15]<[O12] If the Seller wants to support one of the `ProductSpecificationEventType`, then the Seller **MUST** support of all them.

[CR16]<[O10] If the Seller supports the Register for Product Catalog Notification Use Case, the Seller **MUST** support at least one Notification Type. [MEF127 C01]

[R76] The Buyer request **MUST** contain the following attributes [MEF127 R60]:

- `callback`

6.9.1. Register for Event Notifications - Request

To register for notifications the Buyer uses the `registerListener` operation from the API: `POST /hub`. The request model 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.

By using a simple request:

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

The Buyer subscribes for notification of all types of events [O10, O11, O12]

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

```
{
  "callback": "https://buyer.mef.com/listenerEndpoint",
  "query": "eventType=productOfferingCreateEvent,productOfferingStatusChangeEvent"
}
```

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

```
eventType=productOfferingCreateEvent,productOfferingStatusChangeEvent
```

or

```
eventType=productOfferingCreateEvent&eventType=productOfferingStatusChangeEvent
```

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.

[R77] `eventType` is the only attribute that the Seller **MUST** support in the query.

[R78] If the Seller does not support notifications, they **MUST** return an error message to the Buyer indicating that notifications are not supported. [MEF127 R62, MEF127 R61]

[R79] If the Seller does not support any of the notification's `eventType`, they **MUST** return an error message to the Buyer indicating that given `eventType(s)` is not supported, whenever the

Buyer used not supported `eventType(s)` in the `query` attribute of the `EventSubscription`. [MEF127 R62, MEF127 R61]

6.9.2. Register for Event Notifications - Response

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

```
{
  "callback": "https://buyer.mef.com/listenerEndpoint",
  "id": "1659bc83-d334-4de4-aa60-0818e4060ae1",
  "query": "eventType=productOfferingCreateEvent&eventType=productOfferingStatusChangeEvent"
}
```

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

- `https://buyer.mef.com/listenerEndpoint/mefApi/sonata/productCatalogNotifications/v1/listener/productOfferingCreateEvent`

6.9.3. Unregister for Event Notifications - Request

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

[R80] The Buyer must provide the `id` of the registered `EventSubscription` that originates from the Seller.

The example below shows an exemplary unregister call sent by the Buyer to the Seller:

```
http://mef.com:8080/mefApi/sonata/productCatalog/v1/hub/1659bc83-d334-4de4-aa60-0818e4060ae1
```

6.9.4. Unregister for Event Notifications - Response

[R81] In the successful scenario the Seller **MUST** respond with an empty body and HTTP code `204`.

The Buyer can unregister only the whole `EventSubscription`, regardless of the provided `query`. In the case when the Buyer e.g. resigns from specific types of events (or changes the callback address), the previous `EventSubscription` that includes undesired notification types needs to be removed and replaced by the new `EventSubscription` with adjusted `query` attribute.

Note: The conclusion of the above note is that the Buyer cannot update the existing `EventSubscription`. Every kind of update is done by subscription replacement.

6.10. Use case 8: Send Event Notification

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

[R82] The Seller **MUST** send Notifications for `eventTypes` to Buyers who have registered for them. [MEF127 C02, C03, C04, R63]

[R83] The Seller **MUST NOT** send Notifications for `eventTypes` to Buyers who have not registered for them. [MEF113 R157]

[O20] If the Seller fail to receive an acknowledgment from the Buyer repeatedly then Seller **MAY** mark related `EventSubscription` as corrupted and stop sending notifications. [MEF127 O8]

[CR17]<[O20] If the Seller marked related `EventSubscription` as corrupted then unsent notifications **MUST** be stored as dead-letters for resending purposes.

It's at the Buyer and Seller's discretion how to inform the Buyer that the listener is out of service and how to uncheck corrupted `EventSubscription` when the listener is claimed.

The Figure 21 shows all entities involved in the Notification use cases.

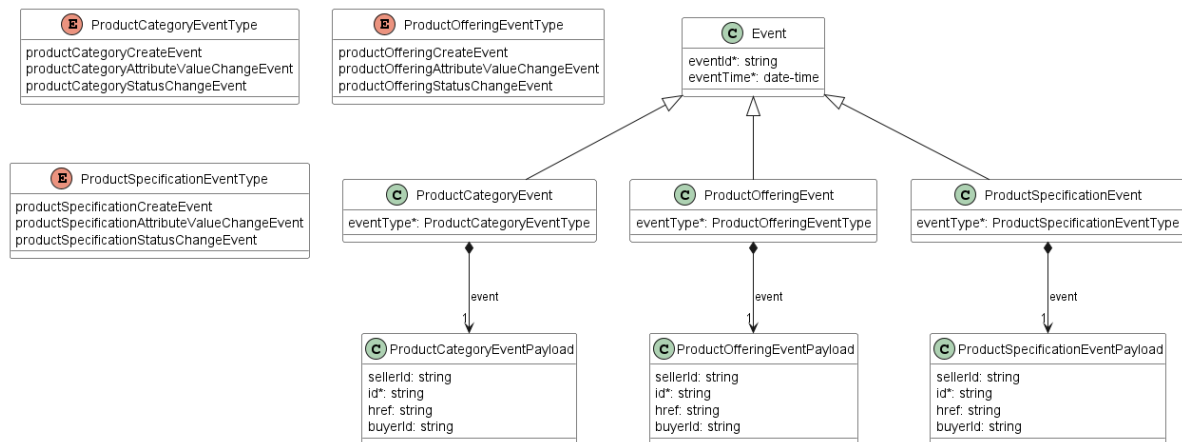


Figure 21: Use Case 8. Notification Data Model

The following snippet presents an example of `productOfferingCreateEvent`

```
{
  "eventId": "f97ec7b4-050d-44b6-91d1-771ad4151695",
  "eventTime": "2023-03-06T15:58:51.968Z",
  "eventType": "productOfferingCreateEvent",
  "event": {
    {
      "id": "b1e4ce38-2328-4d10-8801-f277fbed891d"
    }
  }
}
```

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

The Seller is always the author of any change in the Product Catalog. This implies that each kind of event that is sent by the Seller is triggered by the Seller activities. This is because all the endpoints of the Product Catalog API published to the Buyer are used only for query purposes.

The table below presents the mapping between the API Notification types' names and the ones in MEF 127 together with event descriptions. The inconsistencies are caused by the API naming convention and using the TMF's [TMF620] event types as the base for this API.

API name	MEF 127 name	Description
<code>categoryCreateEvent</code>	PRODUCT_CATEGORY_CREATE	The Seller has published a new Product Category to the Buyers.
<code>categoryAttributeValueChangeEvent</code>	PRODUCT_CATEGORY_UPDATE	The Seller settable attributes for a Product Category were updated by the Seller.
<code>categoryStatusChangeEvent</code>	PRODUCT_CATEGORY_STATE_CHANGE	A Product Category <code>status</code> was changed by the Seller.
<code>productOfferingCreateEvent</code>	PRODUCT_OFFERING_CREATE	The Seller has published a new Product Offering to the Buyers.

API name	MEF 127 name	Description
<code>productOfferingAttributeValueChangeEvent</code>	PRODUCT_OFFERING_UPDATE	The Seller settable attributes for a Product Offering were updated by the Seller.
<code>productOfferingStatusChangeEvent</code>	PRODUCT_OFFERING_STATE_CHANGE	A Product Offering <code>status</code> was changed by the Seller.
<code>productSpecificationCreateEvent</code>	PRODUCT_SPECIFICATION_CREATE	The Seller has published a new Product Specification to the Buyers.
<code>productSpecificationAttributeValueChangeEvent</code>	PRODUCT_SPECIFICATION_UPDATE	The Seller settable attributes for a Product Specification were updated by the Seller.
<code>productSpecificationStatusChangeEvent</code>	PRODUCT_SPECIFICATION_STATE_CHANGE	A Product Specification <code>status</code> was changed by the Seller.

Table 10. Notification types mapping

[R84] The Seller **MUST** include following attributes when sending an event [MEF127 R64]:

- `eventId`
- `eventTime`

- eventType
- event

[CR18]<[O10] The Seller **MUST** send a `categoryCreateEvent` whenever a new Product Category has been created (accordingly to the State Machine described in chapter [6.1.2]) and the Seller supports `createCategoryEvent` event type.

[CR19]<[O10] The Seller **MUST** send a `categoryAttributeValueChangeEvent` whenever the Seller updates any of the following Product Category attributes and supports

`categoryAttributeValueChangeEvent` event type:

- description
- parentCategory
- subCategory
- productOffering

[R85] The Seller **MUST** send a `categoryStatusChangeEvent` whenever a Product Category `status` change occurs (accordingly to the State Machine described in chapter [6.1.2]) and the Seller supports `categoryStatusChangeEvent` event type.

[CR20]<[O11] The Seller **MUST** send a `productOfferingCreateEvent` whenever a new Product Offering has been created (accordingly to the State Machine described in chapter [6.2.2]) and the Seller supports `productOfferingCreateEvent` event type.

[CR21]<[O11] The Seller **MUST** send a `productOfferingAttributeValueChangeEvent` whenever the Seller updates any of the following Product Offering attributes and supports

`productOfferingAttributeValueChangeEvent` event type:

- description
- channel
- agreement
- marketSegment
- region
- category
- statusTransitions
- attachment
- relatedContactInformation
- productOfferingTerm
- note

[R86] The Seller **MUST** send a `productOfferingStatusChangeEvent` whenever a Product Offering `status` change occurs (accordingly to the State Machine described in chapter [6.3.2]) and the Seller supports `productOfferingStatusChangeEvent` event type.

[CR22]<[O12] The Seller **MUST** send a `productSpecificationCreateEvent` whenever a new Product Specification has been created (accordingly to the State Machine described in chapter [6.3.2]) and the Seller supports `productSpecificationCreateEvent` event type.

[CR23]<[O12] The Seller **MUST** send a `productSpecificationAttributeValueChangeEvent` whenever the Seller updates any of the following Product Specification attributes and supports `productSpecificationAttributeValueChangeEvent` event type:

- `description`
- `attachment`
- `note`
- `productSpecificationRelationship` Needs to be checked ???

[R87] The Seller **MUST** send a `productSpecificationStatusChangeEvent` whenever a Product Specification `status` change occurs (accordingly to the State Machine described in chapter [6.3.2]) and the Seller supports `productSpecificationStatusChangeEvent` event type.

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 Product Order 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**.

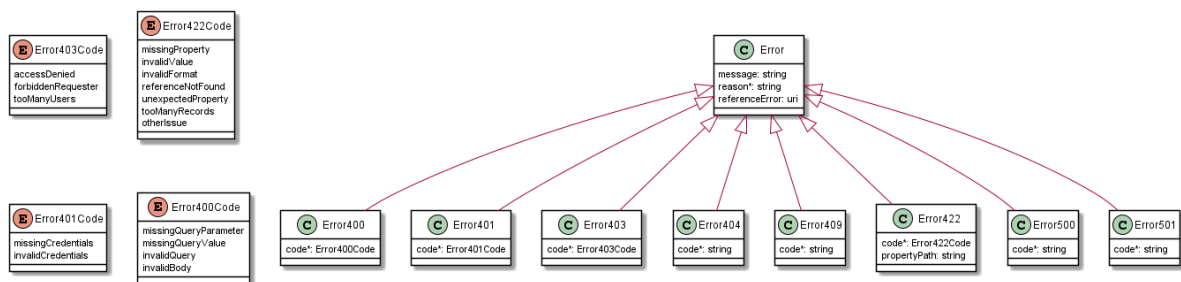


Figure 22. Data model types to represent an erroneous response

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

7.1.1.2. Type Error400

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

Inherits from:

- [Error](#)

Name	Type	Description
------	------	-------------

code*	Error400Code	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	MEF 127
missingQueryParameter	MISSING_QUERY_PARAMETER
missingQueryValue	MISSING_QUERY_VALUE
invalidQuery	INVALID_QUERY
invalidBody	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

Value	MEF 127
missingCredentials	MISSING_CREDENTIALS
invalidCredentials	INVALID_CREDENTIALS

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	MEF 127
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 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.

7.1.1.10. 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

7.1.2. Response pagination

A response to retrieve a list of results (e.g. `GET /productOfferingQualification`) can be paginated. The Buyer 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 Seller 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 Seller 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`

7.2. API Data model

Figure 23 presents the whole Product Catalog data model. The data types, requirements related to them and mapping to MEF 127 specification are discussed later in this section.

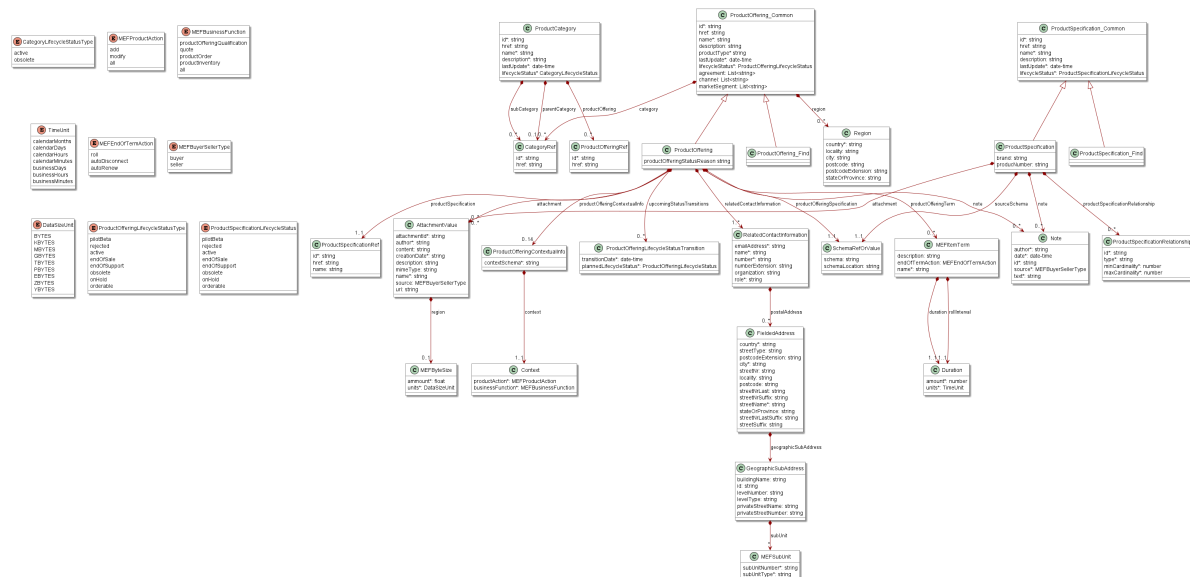


Figure 23. Product Catalog Data Model

7.2.1 Product Category

7.2.1.1 Type ProductCategory

Description: The Product Category is a grouping of Product Offerings in logical containers defined by the Seller. A Product Category may contain other (sub)Product Categories and/or Product Offerings.

Name	Type	M/O	Description	MEF 127
id	string	M	Unique identifier (within the Seller domain) for the Product Category.	Product Category Identifier

Name	Type	M/O	Description	MEF 127
href	uri <i>format = uri</i>	O	Reference of the Product Category	Not represented in MEF 127
name	string	M	The name (unique within the Seller domain) of the Product Category	Product Category Name
description	string	M	Description of the Product Category	Product Category Description
lastUpdate	date-time <i>format = date-time</i>	M	The date and time the Product Category was created or most recently updated.	Product Category Last Update
lifecycleStatus	CategoryLifecycleStatusType	M	The current status of the Product Category.	Product Category State
parentCategory	CategoryRef	O	Identifier referencing the Parent of this Product Category.	Parent Category
subCategory	CategoryRef []	O	A list of references to Product Category, to which this Product Category is a parent of.	Sub Categories
productOffering	ProductOfferingRef []	O	A list of references to Product Offering grouped within this Category	Product Offerings

7.2.1.2 enum [CategoryLifecycleStatusType](#)

Description:

Name	MEF 127 Name	Description
active	AVAILABLE	A Product Category is in the active state when it can be used by the Buyer to retrieve Product Offerings and Product Specifications.
obsolete	OBSOLETE	A Product Category is obsolete when it can no longer be used by the Buyer to retrieve Product Offerings or Product Specifications. The Product Category may be removed from the Product Catalog. This is a final state.
Value	MEF 127	
active	ACTIVE	
obsolete	OBSOLETE	

7.2.1.3 Type CategoryRef

Description: Represents the reference to Category

Name	Type	M/O	Description	MEF 127
id	string	M	Unique (within the Seller domain) identifier for the Category	Not represented in MEF 127
href	uri <small>format = uri</small>	O	Hyperlink to access the Category	Not represented in MEF 127

7.2.1.4 Type ProductOfferingRef

Description: ProductOffering reference. A product offering represents entities that are orderable from the provider of the catalog.

Name	Type	M/O	Description	MEF 127
id	string	M	Unique (within the Seller domain) identifier for the Product Offering.	Not represented in MEF 127
href	uri <small>format = uri</small>	O	Hyperlink to access the Product Offering	Not represented in MEF 127

7.2.2 Product Offering

7.2.2.1 Type ProductOffering_Common

Description: The Product Offering represents the Products orderable from a Seller's Product Catalog.

Name	Type	M/O	Description	MEF 127
id	string	O	Unique identifier (within the Seller domain) for the Product Offering. Note - The Seller must create a new Product Offering Identifier for every new version of a Product Offering. The Seller may choose to incorporate the version information as part of the Offering Identifier.	Product Offering Identifier
href	uri <i>format = uri</i>	O	Hyperlink reference to the Product Offering	Not represented in MEF 127
name	string	O	The commercial name of the Product Offering	Product Offering Name
description	string	O	Description of the Product Offering	Product Offering Description

Name	Type	M/O	Description	MEF 127
lastUpdate	date-time <i>format = date-time</i>	O	The date and time the Product Offering was created or most recently updated.	Product Offering Last Update
lifecycleStatus	ProductOfferingLifecycleStatusType	O		Product Offering State
agreement	string	O	The name of the Seller's offer arrangement (such as a framework agreement). This may be a standard offer agreement or a customer specific agreement (e.g., for a customer specific Product Catalog or customer specific Product Offering). The name is unique within the Seller domain.	Standard Framework Agreement

Name	Type	M/O	Description	MEF 127
channel	string[]	O	<p>The names of the sales channel through which the Product Offering is made available to the Buyer to order. The set of channel names should be specified in the Agreement or provided during the onboarding process. For example: reseller, distribution, directSales.</p> <p>Note: If channel is an empty list, it implies that the Product Offering is available in all Seller supported channels.</p>	Sales Channels

Name	Type	M/O	Description	MEF 127
marketSegment	string[]	O	<p>The names of the market segments targeted for the Product Offering. The set of market segment names should be specified in the Agreement or provided during the onboarding process. For example: wholesale, federal, financial. Note: If marketSegment is an empty list, it implies that the Product Offering is available in all Seller supported market segments</p>	Market Segments

Name	Type	M/O	Description	MEF 127
region	Region[]	O	Areas where the products are offered by the Seller to potential Buyers. Note: If region is an empty list, it implies that the Product Offering is available in all Seller supported Regions.	Regions
category	CategoryRef[]	O	The list of Product Category References in which this Product Offering is grouped together with other Product Offerings available to the Buyer.	Product Offering Product Categories

7.2.2.2 Type ProductOffering

Description: Represents entities that are orderable from the provider of the catalog, this resource included all available information of Product Offering

Inherits from:

- [ProductOffering_Common](#)

Name	Type	M/O	Description
------	------	-----	-------------

Name	Type	M/O	Description
statusTransitions	ProductOfferingLifecycleStatusTransition[]	O	The list of planned Offering Status transitions including the date expected to occur.
productOfferingStatusReason	string	O	Provides complete information on the reason the Product Offering Status is set to a particular value.
attachment	AttachmentValue[]	O	Complements the Product Offering description with presentation, video, etc.
relatedContactInformation	RelatedContactInformation[] <small>minItems = 1</small>	O	Defines the contact role for the related Product Offering. Used to specify the type of contact as specified in MEI Reporter Contact ('role=reporterContact'), REQUIRED in the Buyer Technical Contact ('role=buyerTechnicalContact'), Seller Ticket Contact ('role=sellerTicketContact'), Seller Technical Contact ('role=sellerTechnicalContact').
productOfferingTerm	MEFItemTerm[]	O	Commitment duration for which a Product Offering is available to Buyer. In a Product Offering instance, a Product Offering can be made available for multiple commitment periods of 1, 2 or 3 year terms.
note	Note[]	O	A set of comments providing additional information.

Name	Type	M/O	Description
productSpecification	ProductSpecificationRef	M	A Product Specification Reference to the description of the specifications and defining all of the Offering characteristics.
productOfferingContextualInfo	ProductOfferingContextualInfo[]	O	Defines additional information on the Product Offering Specification for a payload for a Product Offering for each Business and Product Action.
productOfferingSpecification	SchemaRefOrValue	M	Reference (or value) to the Product Offering schema that is consistent with the schema of related Product Offering Specification. Each Product Offering Specification must be valid against the original Product Offering Specification schema (logical subset).

7.2.2.3 Type ProductOffering_Find

Description: Represents entities that are orderable from the provider of the catalog, this resource includes pricing information.

Inherits from:

- [ProductOffering_Common](#)

7.2.2.4 enum ProductOfferingLifecycleStatusType

Description:

Name	MEF 127 Name	Description
------	--------------	-------------

Name	MEF 127 Name	Description
active	ACTIVE	When a Product Offering or Product Specification has been defined and will be made available for ordering; however, it is not yet generally available.
endOfSale	END_OF_SALE	The Product Offering or Product Specification cannot be Installed by any new or existing Buyers, but Buyers may still have Products in use and may Change or Disconnect it, and receive support.
endOfSupport	END_OF_SUPPORT	When a Product Offering or Product Specification in the <code>endOfSale</code> state is no longer supported, the status transitions to <code>endOfSupport</code> . Any existing products can no longer be modified, with the only Order action allowed is <code>delete</code> .
obsolete	OBSOLETE	After a Product Offering or Product Specification that is no longer available it transitions to <code>obsolete</code> and may be removed at the Seller's discretion from the Product Catalog. This is a final state.
onHold	ON_HOLD	A Product Offering or Product Specification that has been <code>orderable</code> , but is currently not available for Buyers due to supply constraints, product recall or other issues preventing it to be offered.
orderable	ORDERABLE	A new Product Offering or Product Specification is in the <code>orderable</code> state when it is available for ordering by Buyers.
inTest	PILOT_BETA	When a Product Offering or Product Specification starts Pilot/Beta testing, it starts in the <code>pilotBeta</code> state .
rejected	REJECTED	When PILOT_BETA testing fails the Product Offering or Product Specification transitions to the <code>rejected</code> state. This is a final state.
Value	MEF 127	
active	ACTIVE	
endOfSale	END_OF_SALE	
endOfSupport	END_OF_SUPPORT	
obsolete	OBSOLETE	
onHold	ON_HOLD	
orderable	ORDERABLE	

Value	MEF 127
inTest	IN_TEST
rejected	REJECTED

7.2.2.5 Type ProductOfferingLifecycleStatusTransition

Description: The planned Product Offering Status transition, including the date it is expected to occur.

Name	Type	M/O	Description	MEF 127
transitionDate	date-time <small>format = date-time</small>	O	The Date and Time that the Next Product Offering Status transition is planned to occur.	Transition Date
transitionLifecycleStatus	ProductOfferingLifecycleStatusType	O	The status of the Product Offering on the planned Transition Date.	Transition Product Offering State

7.2.2.6 Type ProductSpecificationRef

Description: Product Specification reference.

Name	Type	M/O	Description	MEF 127
id	string	M	Unique (within the Seller domain) identifier for the Product Specification.	Not represented in MEF 127
href	uri <small>format = uri</small>	O	Hyperlink to access the Product Specification	Not represented in MEF 127

7.2.2.7 Type MEFItemTerm

Description: The term of the Item

Name	Type	M/O	Description	MEF 127
description	string	O	Description of the term	Description
duration	Duration	M	Duration of the term	Duration
endOfTermAction	MEFEndOfTermAction	M	The action that needs to be taken by the Seller once the term expires	End Of Term Action
name	string	M	Name of the term	Name
rollInterval	Duration	O	The recurring period that the Buyer is willing to pay for the Product after the original term has expired.	Roll Interval

7.2.2.8 enum MEFEndOfTermAction

Description: The action the Seller will take once the term expires. Roll indicates that the Product's contract will continue on a rolling basis for the duration of the Roll Interval at the end of the Term.

Auto-disconnect indicates that the Product will be disconnected at the end of the Term.

Auto-renew indicates that the Product's contract will be automatically renewed for the Term Duration at the end of the Term.

Value	MEF 127
roll	ROLL
autoDisconnect	AUTO_DISCONNECT
autoRenew	AUTO_RENEW

7.2.2.9 Type ProductOfferingContextualInfo

Description: Constrained Product Schema that should be used by the Buyer in the defined Context, where Context is built as pair - a Business Function (e.g. Quote) and Product Action (e.g. install). Those product schemas are created by applying the constraints to Product Schemas defined in the Product Specification. Contextual info MUST be provided for every possible combination of Product Actions and Business Functions (if there are no differences per function of per action then use wildcard - 'all' - and reuse the value of Product Offering Specification attribute).

Name	Type	M/O	Description	MEF 127
------	------	-----	-------------	---------

Name	Type	M/O	Description	MEF 127
contextSchema	SchemaRefOrValue	M	Product Schema that is defined for the given Context. Scheme **MUST** be compliant with JSON scheme standard.	Product Offering Contextual Target Schema
context	Context	M		Not represented in MEF 127

7.2.2.10 Type Context

Description: Context that is defined as a two-dimensional vector of Business Function and Product Action.

Name	Type	M/O	Description	MEF 127
productAction	MEFProductAction	O	Defines Product Action to which the given context applies.	Product Action
businessFunction	MEFBusinessFunction	O	Defines Business Function to which the given context applies.	Business Function

7.2.2.11 enum MEFProductAction

Description: Action that could be applied to the Product (or future product) during the execution of the Business Function. Value 'All' is the wildcard - stands for any action.

Value	MEF 127
add	ADD
modify	MODIFY
all	ALL

7.2.2.12 enum MEFBusinessFunction

Description: Business Function that could be executed for the given Product accordingly to LSO Cantata/Sonata IRPs. Value 'All' is the wildcard - stands for any action.

Value	MEF 127
-------	---------

Value	MEF 127
productOfferingQualification	PRODUCT_OFFERING_QUALIFICATION
quote	QUOTE
productOrder	PRODUCT_ORDER
productInventory	PRODUCT_INVENTORY
all	ALL

7.2.2.13 Type Region

Description: Specifies a region

Name	Type	M/O	Description	MEF 127
locality	string	O	An area of defined or undefined present boundaries within a local authority or other legislatively defined area, usually rural or semi-rural in nature.	Locality
stateOrProvince	string	O	The State or Province that the address is in	State Or Province
country	string	M	The Country the region is located.	Country

7.2.3 Product Specification

7.2.3.1 Type ProductSpecification_Common

Description: Is the basis for all Production Specification representations.

Name	Type	M/O	Description	MEF 127
id	string	O	Unique identifier (within the Seller domain) for the Product Specification.	Product Specification Identifier
href	uri <small>format = uri</small>	O	Reference of the Product Specification	Not represented in MEF 127

Name	Type	M/O	Description	MEF 127
name	string	O	Name of the Product Specification	Product Specification Name
lifecycleStatus	ProductSpecificationLifecycleStatusType	O	The current status of the Product Offering.	Product Specification State
lastUpdate	date-time <small>format = date-time</small>	O	The date and time the Product Specification was created or most recently updated.	Product Specification Last Update

7.2.3.2 Type ProductSpecification

Description: Is a detailed description of a tangible or intangible object made available externally in the form of a ProductOffering to customers or other parties playing a party role.

Inherits from:

- [ProductSpecification_Common](#)

Name	Type	M/O	Description	MEF 1
brand	string	O	The manufacturer or trademark of the Product Specification if the Seller requires a CPE on the Buyer's premise.	Brand
description	string	M	Description of the Product Specification.	Product Specific Descrip

Name	Type	M/O	Description	MEF 1
productNumber	string	O	An identifier assigned to the model or version of the CPE used in conjunction with the Brand (for example hardware SKU or software license key).	Product Number
attachment	AttachmentValue[]	O	Complements the Product Offering description with presentation, video, pictures, etc. This would only be expected to be used to provide additional information if there is a CPE re-quired, for instance a link to the website of the CPE vendor.	Product Specific Attachr
note	Note[]	O	A set of comments for additional information.	Product Specific Notes

Name	Type	M/O	Description	MEF 1
sourceSchema	SchemaRefOrValue	M	A reference to the schema that defines the Product.	Source Product Specific Schema
productSpecificationRelationship	ProductSpecificationRelationship[]	M	List of relations between defined Product Specifications.	Product Specification Relationship

7.2.3.3 Type ProductSpecification_Find

Description: Is a lightweight version of the Product Specification object used in Get List use case.

Inherits from:

- [ProductSpecification_Common](#)

7.2.3.4 enum ProductSpecificationLifecycleStatusType

Description:

Name	MEF 127 Name	Description
active	ACTIVE	When a Product Specification has been defined and will be made available for ordering; however, it is not yet generally available.
endOfSale	END_OF_SALE	The endOfSale status means that Product Specification cannot be used for creation of a new Product Offerings.
endOfSupport	END_OF_SUPPORT	When a Product Specification in the endOfSale status is no longer supported, the status transitions to endOfSupport. Any existing products can no longer be Changed, with the only Order action allowed is Disconnect.
obsolete	OBSOLETE	After a Product Specification that is no longer available it transitions to obsolete and may be removed at the Seller's discretion from the Product Catalog. This is a final state.

Name	MEF 127 Name	Description
onHold	ON_HOLD	A Product Specification that has been orderable, but is currently not available for Buyers due to supply constraints, product recall or other issues preventing it to be offered.
orderable	ORDERABLE	A Product Specification is in the orderable state when it is available for ordering by Buyers.
inTest	PILOT_BETA	When a Product Specification starts Pilot/Beta testing, it starts in the inTest state.
rejected	REJECTED	When Pilot/Beta testing fails the Product Specification to the rejected state. This is a final state.
Value	MEF 127	
active	ACTIVE	
endOfSale	END_OF_SALE	
endOfSupport	END_OF_SUPPORT	
obsolete	OBSOLETE	
onHold	ON_HOLD	
orderable	ORDERABLE	
inTest	IN_TEST	
rejected	REJECTED	

7.2.3.5 Type ProductSpecificationRelationship

Description: Uni-directional relationship between the Products (the owner of the relation is the Product that defines it). Relation defines the type and the cardinalities.

Name	Type	M/O	Description	MEF 127
id	string	M	Identifier of the targeted Product Specification	Related Product Specification Identifier
relationshipType	string	O	Defines the type of the relation or the role of the relation owner in the context of this relation e.g. hosts, aggregates	Relationship Type

Name	Type	M/O	Description	MEF 127
minCardinality	integer <i>default = 0</i> <i>minimum = 0</i>	O	The minimal number of the related Products that must be satisfied. e.g. Access E-Line requires at least two End Points.	Min Cardinality

maxCardinality	integer <i>default = -1</i> <i>maximum = -1</i>	O	The maximal number of the related Products that must be satisfied. e.g. Endpoint must be related with only one Access E-Line. `-1` stands for infinity i.e. any number of instances of the given type could be related to the considered instance.	Max Cardinality
----------------	---	---	--	-----------------

7.2.4 Common types

This chapter includes common types that are shared between Product Category, Product Offering, or Product Specification types.

7.2.4.1 Type AttachmentValue

Description: Complements the description of an element (for instance a product) through video, pictures...

Name	Type	M/O	Description	MEF 127
attachmentId	string	O	locally unique identifier to distinguish items from the Attachment list.	Not represented in MEF 127
author	string	M	The name of the person or organization who added the Attachment.	Attachment Author
content	string	O	The actual contents of the attachment object, if embedded, encoded as base64. Either url or (content and mimeType) attributes MUST be provided during creation.	Content
creationDate	date-time <i>format = date-time</i>	M	The date the Attachment was added.	Attachment Date

Name	Type	M/O	Description	MEF 127
description	string	O	A narrative text describing the content of the attachment	Description
contentType	string	O	Attachment mime type such as extension file for video, picture and document	Mime Type
name	string	M	The name of the attachment	Attachment Name
size	MEFByteSize	O	The size of the attachment.	Size
source	MEFBuyerSellerType	M	Indicates if the attachment was added by the Buyer or the Seller.	Attachment Source
url	string	O	URL where the attachment is located. Either url or (content and mimeType) attributes MUST be provided during creation.	URL

7.2.4.2 Type MEFByteSize

Description: A size represented by value and Byte units

Name	Type	M/O	Description	MEF 127
amount	float <small>default = 1</small> <small>format = float</small>	M	Numeric value in a given unit	Value
units	DataSizeUnit	M	Byte Unit	Unit

7.2.4.3 enum DataSizeUnit

Description: The unit of measure in the data size.

Value	MEF 127
BYTES	BYTES
KBYTES	KBYTES
MBYTES	MBYTES
GBYTES	GBYTES
TBYTES	TBYTES

Value	MEF 127
PBYTES	PBYTES
EBYTES	EBYTES
ZBYTES	ZBYTES
YBYTES	YBYTES

7.2.4.4 enum MEFBuyerSellerType

Description: An enumeration with buyer and seller values.

Value	MEF 127
buyer	BUYER
seller	SELLER

7.2.4.5 Type RelatedContactInformation

Description: Contact data for a person or organization that is involved in a given context. It is specified by the Seller (e.g. Seller Contact Information) or by the Buyer.

Name	Type	M/O	Description	MEF 127
emailAddress	string	M	Email address	Email address
name	string	M	Name of the contact	Name of the contact
number	string	M	Phone number	Contact Phone Number
numberExtension	string	O	Phone number extension	Contact Phone Number Extension
organization	string	O	The organization or company that the contact belongs to	Contact Organization
postalAddress	FieldedAddress	O	Identifies the postal address of the person or office to be contacted.	Contact Postal Address
role	string	M	A role the party plays in a given context.	Not represented in MEF 127

7.2.4.6 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)

Name	Type	M/O	Description	MEF 127
country	string	M	Country that the address is in	Country
streetType	string	O	The type of street (e.g., alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf)	Street Type
postcodeExtension	string	O	An extension of a postal code. E.g. the part following the dash in a US urban property address	Postal Code Extension
city	string	M	The city that the address is in	City
streetNr	string	O	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.	Street Number

Name	Type	M/O	Description	MEF 127
locality	string	O	The locality that the address is in	Locality
postcode	string	O	Descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as zip code)	Postal Code
streetNrLast	string	O	Last number in a range of street numbers allocated to a property	Street Number Last
streetNrSuffix	string	O	The first street number suffix	Street Number Suffix
streetName	string	M	Name of the street or other street type	Street Name
stateOrProvince	string	O	The State or Province that the address is in	State Or Province
streetNrLastSuffix	string	O	Last street number suffix for a ranged address	Street Number Last Suffix
geographicSubAddress	GeographicSubAddress	O		Not represented in MEF 127
streetSuffix	string	O	A modifier denoting a relative direction	Street Suffix

7.2.4.7 Type GeographicSubAddress

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

Name	Type	M/O	Description	MEF 127
------	------	-----	-------------	---------

Name	Type	M/O	Description	MEF 127
buildingName	string	O	Allows for identification of places that require building name as part of addressing information	Building Name
id	string	O	Unique Identifier of the subAddress	Not represented in MEF 127
levelNumber	string	O	Used where a level type may be repeated e.g. BASEMENT 1, BASEMENT 2	Level Number
levelType	string	O	Describes level types within a building	Level Type
privateStreetName	string	O	"Private streets internal to a property (e.g. a university) may have internal names that are not recorded by the land title office	Private Street Name
privateStreetNumber	string	O	Private streets numbers internal to a private street	Private Street Number
subUnit	MEFSubUnit[]	O	Representation of a MEFSubUnit It is used for describing subunit within a subaddress e.g.BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.	Not represented in MEF 127

7.2.4.8 Type MEFSubUnit

Description: Allows for sub unit identification

Name	Type	M/O	Description	MEF 127
subUnitNumber	string	M	The discriminator used for the subunit, often just a simple number but may also be a range.	Sub Unit Number

Name	Type	M/O	Description	MEF 127
subUnitType	string	M	The type of subunit e.g.BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.	Sub Unit Type

7.2.4.9 Type Note

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

Name	Type	M/O	Description	MEF 127
author	string	M	Author of the note	Note Author
date	date-time <small>format = date-time</small>	M	Date the Note was created	Note Date
id	string	M	Identifier of the note within its containing entity (may or may not be globally unique, depending on provider implementation)	Not represented in MEF 127
source	MEFBuyerSellerType	M	Indicates if the note is from Buyer or Seller	Note Source
text	string	M	Text of the note	Note Text

7.2.4.10 Type SchemaRefOrValue

Description: Reference to the JSON schema location or the exact value of the JSON schema. **Note:** One of the properties must be provided i.e. schemaLocation or schema.

Name	Type	M/O	Description	MEF 127
schema	string	O	Raw JSON schema value.	Not represented in MEF 127
schemaLocation	uri <small>format = uri</small>	O	This field provides a link to the schema describing the target product	Not represented in MEF 127

7.2.4.11 Type Duration

Description: A Duration in a given unit of time e.g. 3 hours, or 5 days.

Name	Type	M/O	Description	MEF 127
amount	integer	M	Duration (number of seconds, minutes, hours, etc.)	Value
units	TimeUnit	M	Time unit type	Unit

7.2.4.12 enum TimeUnit

Description: Represents a unit of time. Reference: MEF 57.2 (Sn 9.22)

Value	MEF 127
calendarMonths	CALENDAR_MONTHS
calendarDays	CALENDAR_DAYS
calendarHours	CALENDAR_HOURS
calendarMinutes	CALENDAR_MINUTES
businessDays	BUSINESS_DAYS
businessHours	BUSINESS_HOURS
businessMinutes	BUSINESS_MINUTES

7.2.5 Notification Registration

Notification registration and management are done through [/hub](#) API endpoint. The below sections describe data models related to this endpoint.

7.2.8.1. Type EventSubscriptionInput

The query attribute is used to constrain the notification types that the Buyer is willing to receive to the callback endpoint. The query formatting complies to RCF3986 [rfc3986](#) and [TMF620](#). Every attribute defined in the Event model (from notification API) can be used in the query. Example:

```
"query": "eventType=productOfferingCreateEvent"
```

If the Buyer wishes to subscribe to 2 different types of events, there are 2 possible syntax variants:

- `eventType=productOfferingCreateEvent,productOfferingStatusChangeEvent` **or**
- `eventType=productOfferingCreateEvent&eventType=productOfferingStatusChangeEvent`

Description: This class is used to register for Notifications.

Name	Type	M/O	Description
callback	string	M	This callback value must be set to <code>*host*</code> property from Buyer Product Catalog (productCatalogNotification.api.yaml). This property is appended with the base specified in that API to construct an URL to which notification is sent. E.g. for <code>"https://buyer.com/listenerEndpoint"</code> , the Product Specification state change event <code>https://buyer.com/listenerEndpoint/mefApi/sonata/productCatalogSpecification</code>
query	string	O	This attribute is used to define to which type of events to register to. See the <code>'ProductSpecificationEventType'</code> , <code>'ProductOfferingEventType'</code> in (productCatalogNotification.api.yaml) for the kind of events are supported. To subscribe for more than one event type, put the <code>'eventType=productOfferingCreateEvent,productOfferingAttributeValueChange'</code> or <code>'eventType=productOfferingCreateEvent&eventType=productOfferingAttribute'</code> treated as specifying no filters - ending in subscription for all event types.

7.2.8.2. Type EventSubscription

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

Name	Type	M/O	Description	MEF 127
callback	string	M	The value provided by the Buyer in <code>'EventSubscriptionInput'</code> during notification registration	
id	string	M	An identifier of this Event Subscription assigned by the Seller when a resource is created.	
query	string	O	The value provided by the Buyer in <code>'EventSubscriptionInput'</code> during notification registration	

7.3. Notification API Data model

Figure 24 presents the Product Catalog Notification data model. section.

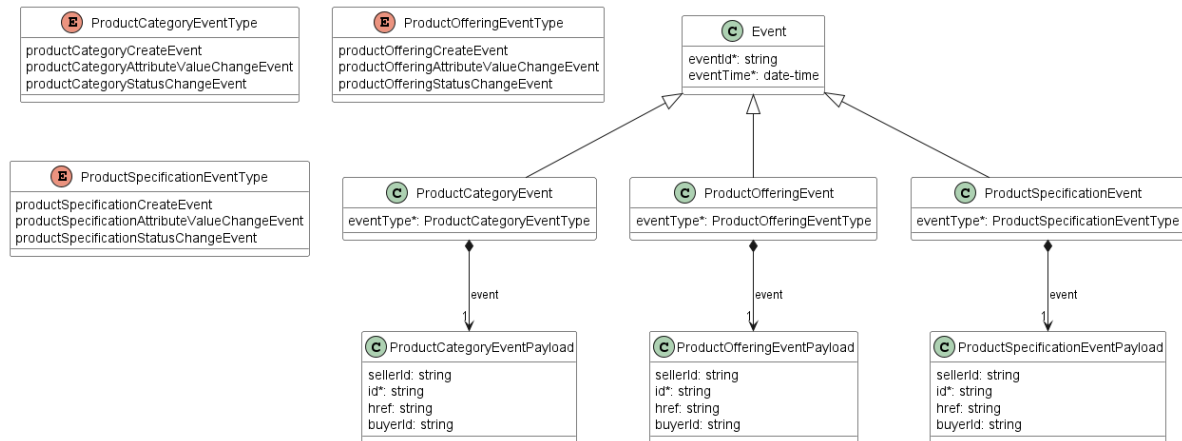


Figure 24. Product Catalog Notification Notification Data Model

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	M/O	Description	MEF 127
eventId	string	M	Id of the event	
eventTime	date-time <i>format = date-time</i>	M	Datetime when the event occurred	

7.3.2. Type ProductCategoryEvent

Description:

Inherits from:

- [Event](#)

Name	Type	M/O	Description	MEF 127
eventType	ProductCategoryEventType	M	Indicates the type of the event.	
event	ProductCategoryEventPayload	M	A reference to the object that is source of the notification.	

7.3.3. Type ProductCategoryEventPayload

Description: The identifier of the Product Category being subject of this event.

Name	Type	M/O	Description	MEF 127
sellerId	string	O	The unique identifier of the organization that is acting as the Seller. MUST be specified in the request only when requester entity represents more than one Seller.	
id	string	M	ID of the Product Category attributed by the Seller	
href	uri <small>format = uri</small>	O	Hyperlink to access the Product Category	
buyerId	string	O	The unique identifier of the organization that is acting as the a Buyer. MUST be specified in the request only when the responding represents more than one Buyer.	

7.3.4. enum ProductCategoryEventType

Description: Type of the Product Category event.

API name	MEF 127 name	Description
<small>categoryCreateEvent</small>	PRODUCT_CATEGORY_CREATE	The Seller has published new Product Category to the Buyers.
<small>categoryAttributeValueChangeEvent</small>	PRODUCT_CATEGORY_UPDATE	The Seller settable attributes for a Product Category were updated by the Seller.
<small>categoryStateChangeEvent</small>	PRODUCT_CATEGORY_STATE_CHANGE	A Product Category <small>status</small> was changed by the Seller.
Value	MEF 127	
categoryCreateEvent	CATEGORY_CREATE_EVENT	

Value	MEF 127
categoryAttributeValueChangeEvent	CATEGORY_ATTRIBUTE_VALUE_CHANGE_EVENT
categoryStateChangeEvent	CATEGORY_STATE_CHANGE_EVENT

7.3.5. Type ProductOfferingEvent

Description:

Inherits from:

- [Event](#)

Name	Type	M/O	Description	MEF 127
eventType	ProductOfferingEventType	M	Indicates the type of the event.	
event	ProductOfferingEventPayload	M	A reference to the object that is source of the notification.	

7.3.6. Type ProductOfferingEventPayload

Description: The identifier of the Product Offering being subject of this event.

Name	Type	M/O	Description	MEF 127
sellerId	string	O	The unique identifier of the organization that is acting as the Seller. MUST be specified in the request only when requester entity represents more than one Seller.	
id	string	M	ID of the Product Offering attributed by the Seller	
href	uri <small>format = uri</small>	O	Hyperlink to access the Product Offering	
buyerId	string	O	The unique identifier of the organization that is acting as the a Buyer. MUST be specified in the request only when the responding represents more than one Buyer.	

7.3.7. [enum](#) ProductOfferingEventType

Description: Type of the Product Offering event.

API name	MEF 127 name	Description
----------	--------------	-------------

API name	MEF 127 name	Description
<code>productOfferingCreateEvent</code>	PRODUCT_OFFERING_CREATE	The Seller has published new Product Offering to the Buyers.
<code>productOfferingAttributeValueChangeEvent</code>	PRODUCT_OFFERING_UPDATE	The Seller settable attributes for a Product Offering were updated by the Seller.
<code>productOfferingStateChangeEvent</code>	PRODUCT_OFFERING_STATE_CHANGE	A Product Offering <code>status</code> was changed by the Seller.

Value	MEF 127
<code>productOfferingCreateEvent</code>	PRODUCT_OFFERING_CREATE_EVENT
<code>productOfferingAttributeValueChangeEvent</code>	PRODUCT_OFFERING_ATTRIBUTE_VALUE_CHAN
<code>productOfferingStateChangeEvent</code>	PRODUCT_OFFERING_STATE_CHANGE_EVENT

7.3.8. Type ProductSpecificationEvent

Description:

Inherits from:

- [Event](#)

Name	Type	M/O	Description	MEF 127
<code>eventType</code>	ProductSpecificationEventType	M	Indicates the type of the event.	

Name	Type	M/O	Description	MEF 127
event	ProductSpecificationEventPayload	M	A reference to the object that is source of the notification.	

7.3.9. Type ProductSpecificationEventPayload

Description: The identifier of the Product Specification being subject of this event.

Name	Type	M/O	Description	MEF 127
id	string	M	ID of the Product Specification attributed by the Seller	
href	uri <small>format = uri</small>	O	Hyperlink to access the Product Specification	
buyerId	string	O	The unique identifier of the organization that is acting as the a Buyer. MUST be specified in the request only when the responding represents more than one Buyer.	
sellerId	string	O	The unique identifier of the organization that is acting as the Seller. MUST be specified in the request only when requester entity represents more than one Seller.	

7.3.10. enum ProductSpecificationEventType

Description: Type of the Product Specification event.

API name	MEF 127 name	Description
productSpecificationCreateEvent	PRODUCT_SPECIFICATION_CREATE	The Seller has published new Product Specification to the Buyers.

API name	MEF 127 name	Description
<code>productSpecificationAttributeValueChangeEvent</code>	PRODUCT_SPECIFICATION_UPDATE	The Seller settable attributes for a Product Specification were updated by the Seller.
<code>productSpecificationStateChangeEvent</code>	PRODUCT_SPECIFICATION_STATE_CHANGE	A Product Specification <code>status</code> was changed by the Seller.

Value	MEF 127
<code>productSpecificationCreateEvent</code>	PRODUCT_SPECIFICATION_CREATE_EVENT
<code>productSpecificationAttributeValueChangeEvent</code>	PRODUCT_SPECIFICATION_ATTRIBUTE_VALU
<code>productSpecificationStateChangeEvent</code>	PRODUCT_SPECIFICATION_STATE_CHANGE_I

8. References

- [OAS-v3] [Open API 3.0](#), February 2020
- [MEF55.1] [MEF 55.1](#), Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, February 2021
- [MEF79.1] [MEF 79.1](#), Address, Service Site, and Product Offering Qualification Management, Requirements and Use Cases, November 2019
- [MEF57.2] [MEF 57.2](#), Product Order Management Business Requirements and Use Cases, October 2022
- [MEF127] [MEF 127] (<https://www.mef.net/wp-content/uploads/MEF-127-Draft-R1.pdf>), Product Catalog Business Requirements and Use Cases, February 2023, Draft Standard (R1)
- [MEF128] [MEF 128] (<https://www.mef.net/wp-content/uploads/MEF-128.pdf>), LSO API Security Profile, July 202
- [REST] [Chapter 5: Representational State Transfer \(REST\)](#) Fielding, Roy Thomas, Architectural Styles and the Design of Network-based Software Architectures (Ph.D.).
- [RFC2119] [RFC 2119](#), Key words for use in RFCs to Indicate Requirement Levels, March 1997
- [RFC3986] [RFC 3986](#) Uniform Resource Identifier (URI): Generic Syntax, January 2005
- [RFC8174] [RFC 8174](#), Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, May 2017
- [TMF620] [TMF 620](#), Product Catalog API REST Specification 4.1.0, April 2021
- [TMF630] [TMF 630](#) TMF630 API Design Guidelines 4.2.0
- [JSONSchema] [JSON Schema](#) JSON Schema main page