MEF Standard
MEF 87

LSO Cantata and LSO Sonata Product Offering Qualification API – Developer Guide

May 2022

Table of Contents

List of Figures

List of Tables

1 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.

- Amartus
- Cisco
- Colt
- Lumen Technologies
- NEC/Netcracker
- Orange
- Proximus
- Spirent Communications
- Verizon

2 Abstract

This standard is intended to assist implementation of the Product Offering Qualification (POQ) function defined for the LSO Cantata and LSO Sonata Interface Reference Points (IRPs), for which requirements and use cases are defined in MEF 79 [7], MEF 79.0.1 [10], and MEF 79.0.2 [11]. This standard consists of this document and complementary API definitions for Product Offering Qualification Management and Notification.

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

commit id: 2062c16db194adc5109d0b7c0578a1a9128c6471

- `productApi/serviceability/offeringQualification/productOfferingQualificationManagement.api.yaml`
- `productApi/serviceability/offeringQualification/productOfferingQualificationNotification.api.yaml`

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

commit id: fd4aad8d6417b6aed2fa4e2d4ffa9836648addb0

- `productApi/serviceability/offeringQualification/productOfferingQualificationManagement.api.yaml`
- `productApi/serviceability/offeringQualification/productOfferingQualificationNotification.api.yaml`

3 Terminology and Abbreviations

This section defines the terms used in this document. In many cases, the normative definitions to 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.

Term	Definition	Reference
Application Program Interface (API)	In the context of LSO, API describes one of the Management Interface Reference Points based on the requirements specified in an Interface Profile, along with a data model, the protocol that defines operations on the data and the encoding format used to encode data according to the data model. In this document, API is used synonymously with REST API.	MEF 55.1 [8]
Buyer	In the context of this document, denotes the organization or individual acting as the customer in a transaction over a Cantata (Customer ↔ Service Provider) or Sonata (Service Provider ↔ Partner) Interface.	This document, adapted from MEF 80 [12]
Deferred Response	A Seller's response to a Buyer's request whereby the Seller immediately acknowledges that the request was received, and, over time, sends notifications to update the Buyer on the status and results of the request (assuming the Buyer has subscribed to receive the notifications). The Buyer can also poll the Seller for the results and status associated with the request.	MEF 79 [9]
Immediate Response	A Seller's response to the Buyer whereby the Seller responds immediately with the results of the request or indicates that the request cannot be processed. The maximum time to provide an Immediate Response is for further study, but is expected to be less than 30 seconds.	MEF 79 [9]
OpenAPI	RESTful API Documentation Specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services.	OAS v3.0.3 [13]
POQ	Product Offering Qualification	MEF 79 [9]
POQ Item	Product Offering Qualification Item	MEF 79 [9]
Product Offering Qualification	One or more POQ Items formulated into a request made by a Buyer to a Seller.	MEF 79 [9]

Term	Definition	Reference
Product Offering Qualification Item	An individual article included in a POQ that describes a Product of a particular type (Product Offering). The objective is to determine if it is feasible for the Seller to deliver this item as described and for the Seller to inform the Buyer of the estimated time interval to complete this delivery.	MEF 79 [9]
Requesting Entity	The business organization that is acting on behalf of one or more Buyers. In the most common case, the Requesting Entity represents only one Buyer and these terms are then synonymous.	MEF 79 [9]
Responding Entity	The business organization that is acting on behalf of one or more Sellers. In the most common case, the Responding Entity represents only one Seller and these terms are then synonymous.	MEF 79 [9]
Representational State Transfer Application Program Interface	REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems.	Fielding [7]
REST API	Representational State Transfer Application Program Interface	Fielding [7]
Seller	In the context of this document, denotes the organization acting as the supplier in a transaction over a Cantata (Customer ↔ Service Provider) or Sonata (Service Provider ↔ Partner) Interface.	This document; adapted from MEF 80 [12]

Table 1 – Terminology and Abbreviations

4 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 [2], RFC 8174 [6]) 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**.**

5 Introduction

This standard specification document describes the Application Programming Interface (API) for Product Offering Qualification functionality of the LSO Cantata Interface Reference Point (IRP) and LSO Sonata IRP as defined in the MEF 55.1 *Lifecycle Service Orchestration (LSO): Reference Architecture and Framework* [8]. The LSO Reference Architecture is shown in Figure 1 with both IRPs highlighted.

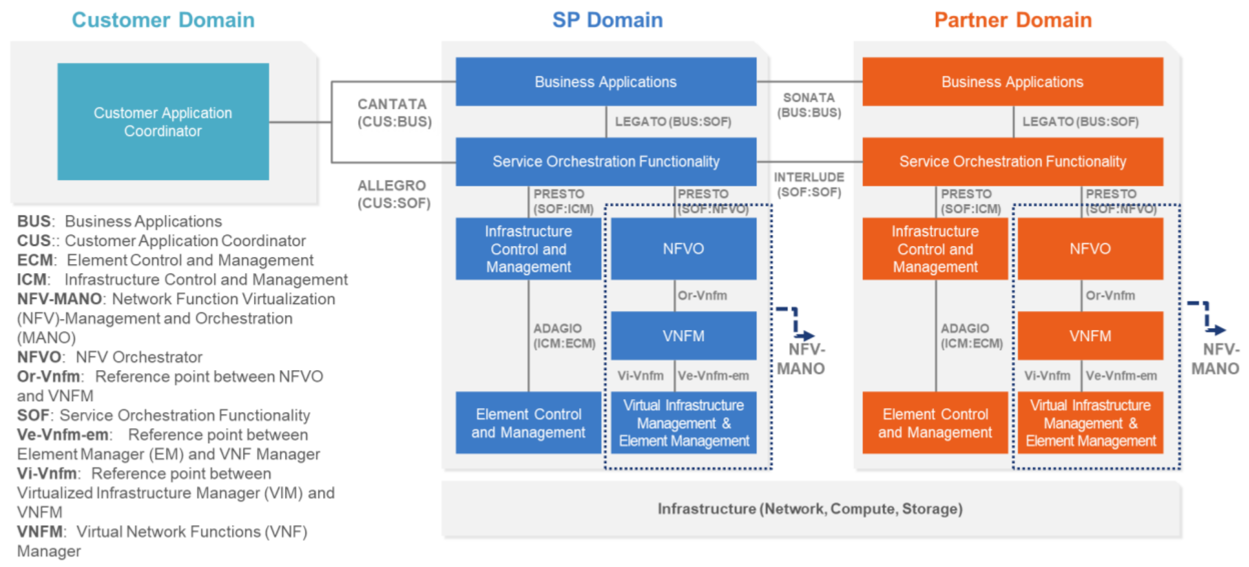
Customer Domain
SP Domain
Partner Domain
Customer Application Coordinator
CANTATA (CUS:BUS)
ALLEGRO (CUS:SOF)
Business Applications
LEGATO (BUS:SOF)
Service Orchestration Functionality
SONATA (BUS:BUS)
INTERLUDE (SOF:SOF)
PRESTO (SOF:ICM)
PRESTO (SOF:NFVO)
Infrastructure Control and Management
NFVO
Or-Vnfm
ADAGIO (ICM:ECM)
VNFM
Vi-Vnfm
Ve-Vnfm-em
Element Control and Management
Virtual Infrastructure Management & Element Management
NFV-MANO
Infrastructure (Network, Compute, Storage)
BUS: Business Applications
CUS:: Customer Application Coordinator
ECM: Element Control and Management
ICM: Infrastructure Control and Management
NFV-MANO: Network Function Virtualization (NFV)-Management and Orchestration (MANO)
NFVO: NFV Orchestrator
Or-Vnfm: Reference point between NFVO and VNFM
SOF: Service Orchestration Functionality
Ve-Vnfm-em: Reference point between Element Manager (EM) and VNF Manager
Vi-Vnfm: Reference point between Virtualized Infrastructure Manager (VIM) and VNFM
VNFM: Virtual Network Functions (VNF) Manager

Figure 1 – The LSO Reference Architecture

Cantata and Sonata IRPs define pre-ordering and ordering operations that allow 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 Offering Qualification are defined in:

- MEF 79: Address, Service Site, and Product Offering Qualification Management – Requirements and Use Cases [9],

- MEF 79.0.1, Amendment to MEF 79: Address, Service Site, and Product Offering Qualification Management Requirements and Use Cases [10], and
- MEF 79.0.2: *Amendment to MEF 79: Address Validation* [11].

This document focuses on implementation aspects of POQ functionality and is structured as follows:

- Section 5 gives a technical introduction to POQ functionality.
- Section 6 provides an overview of the operations, data models, and design patterns of API definition.
- Section 7 focuses on API interactions with the help of end-to-end sequence diagrams and usage examples.
- Section 8 complements Section 6 with an in-depth API description.

5.1 Description

The Product Offering Qualification (POQ) API allows a 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 service site specified by the Buyer; or modify a previously purchased product.

The API payloads exchanged between a Buyer and the Seller during the POQ execution consist of product-independent and product-specific parts. The product-independent part is technically defined in this standard. The product-specific part is defined in the product specifications of the concerned product. Both definition types must be used in combination to validate the correctness of the payloads. Section 6.2.3 explains how to use product-specific definitions with POQ API definition.

This document uses samples of Access E-Line Product specification definitions to construct API payload examples in Section 7.

Note: The Access E-Line product is valid only in the Sonata context. It is used only for the explanation of the rules of combining the product-agnostic (envelope) and product-specific (payload) parts of the APIs. The examples are not normative and are not updated to reflect new versions of the product specification (MEF 106). It is out of the scope of this document to explain the details of any product.

Product specifications are defined using the JSON Schema (draft 7) standard [1], whereas POQ API is defined using the OpenAPI 3.0 standard [13]. The payloads exchanged through POQ endpoints must comply with these definitions as well as with MEF 79 [9], MEF 79.0.1 [10], and MEF 79.0.2 [11] requirements for POQ functionality and with product specifications that are in use.

5.2 Conventions in this 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. `GeographicAddress`).
- In UML diagrams the default cardinality of associations is `0..1`. Other cardinality markers are complaint with the UML standard.
- In the API details tables and UML diagrams required attributes are marked with a `*` next to their names.
- In UML sequence diagrams `{{variable}}` notation is used to indicate a variable to be substituted with a correct value.

5.3 Relation to Other Documents

The requirements and use cases for POQ functionality are defined in MEF 79 [9], MEF 79.0.1 [10], and MEF 79.0.2 [11]. The API definition builds on TMF 679 version 19.0.1 [14]. POQ functions must allow using any MEF product specifications.

5.4 Approach

As presented in Figure 2, both Cantata and Sonata API frameworks consists 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)

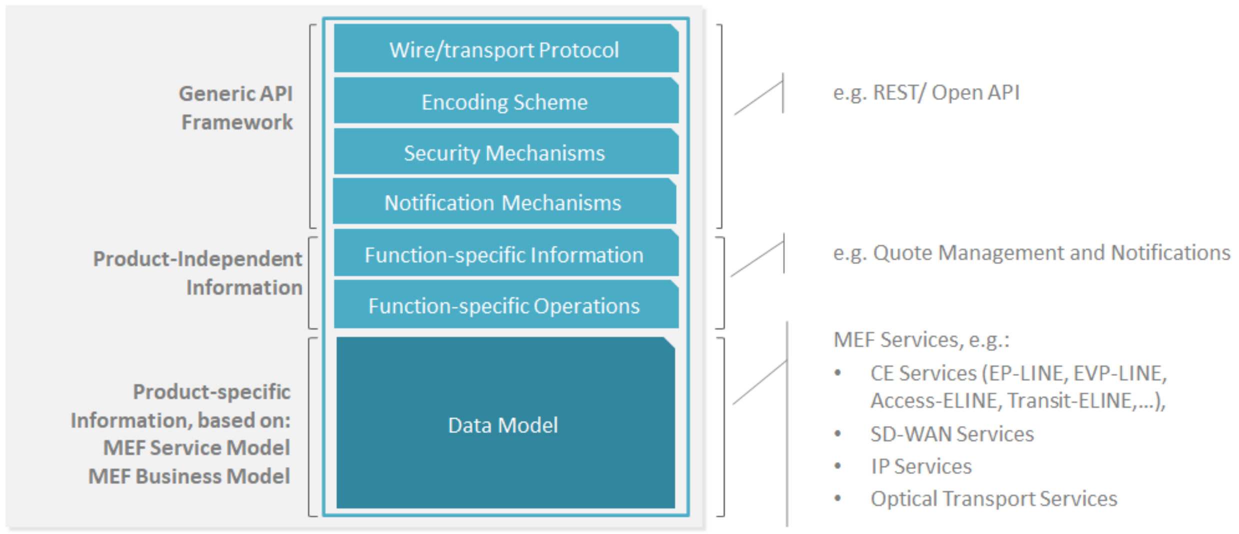
Generic API Framework
Product-Independent Information
Product-specific Information, based on: MEF Service Model MEF Business Model
Wire/transport Protocol
Encoding Scheme
Security Mechanisms
Notification Mechanisms
Function-specific Information
Function-specific Operations
Data Model
e.g. REST/ Open API
e.g. Quote Management and Notifications
MEF Services, e.g.:
CE Services (EP-LINE, EVP-LINE, Access-ELINE, Transit-ELINE,…),
SD-WAN Services
IP Services
Optical Transport Services

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 part of the framework focuses on a model of a particular Cantata or Sonata functionality and is agnostic to any of the product specifications. For example, this standard is describing Product Offering Qualification model and operations that allow performing qualifications of any product that is aligned with one of MEF or custom 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. This standard is not defining MEF product specifications, however, can be used along with any product specification defined by or compliant with MEF.

5.5 High-Level Flow

Product Offering Qualification is part of a broader Cantata and Sonata End-to-End flow. Figure 3 below shows a high-level diagram to get a good understanding of the whole process and Product Offering Qualification's position within it.

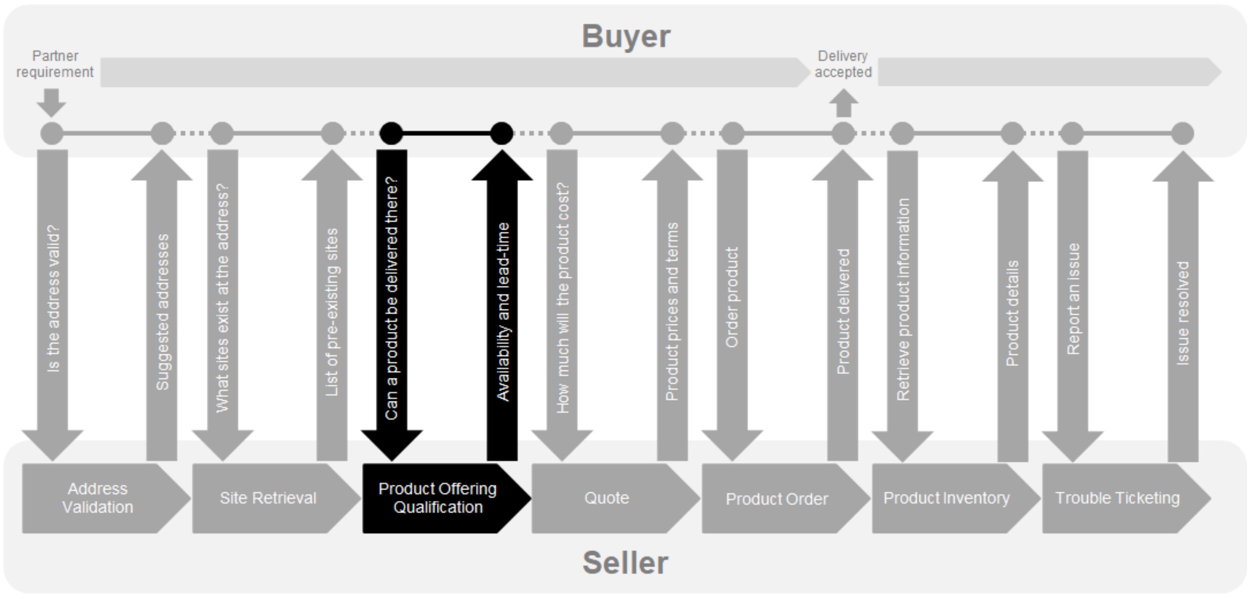

Figure 3 – Cantata and Sonata End-to-End Function Flow

- 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 Service Site information including exact formats for Service 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 service 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 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 that are not part of normal operations of the Product provided by the Seller.

6 API Description

This section discusses the API structure and design patterns. It starts with a description of available REST endpoints. Then, an overview of the API data model is given together with the description of the extension design pattern that is used to combine product-agnostic and product-specific parts of API payloads. Finally, payload validation and API security aspects are discussed.

6.1 Resource/Endpoint Description

6.1.1 Seller Side Endpoints

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

Base URL for Sonata:
`https://{{serverBase}}:{{port}}{{?/seller_prefix}}/mefApi/sonata/productOfferingQualification/v7/`

The following API endpoints are implemented by the Seller and allow the Buyer to send POQ requests, retrieve existing POQs or POQ details, and manage notification registrations. The endpoints and corresponding data model are defined in `productApi/serviceability/offeringQualification/productOfferingQualificationManagement.api.yaml`.

API Endpoint	**Description**	**MEF 79 Use Case Mapping**
`POST /productOfferingQualification`	A request initiated by the Buyer to determine whether the Seller can feasibly deliver a particular Product (or Products) to a specific set of geographic locations.	Use Case 6: Create Product Offering Qualification
`GET /productOfferingQualification`	A request initiated by the Buyer to retrieve a list of POQs (in any state) from the Seller based on a set of POQ filter criteria.	Use Case 7: Retrieve POQ List

GET /productOfferingQualification/{{id}}	A request initiated by the Buyer to retrieve full details of a single Product Offering Qualification based on a POQ identifier.	Use Case 8: Retrieve POQ by Identifier
POST /hub	A request initiated by the Buyer to instruct the Seller to send notifications of POQ create and/or state changes if the Buyer has registered for these notifications. The state change notifications are sent only in the Deferred scenario as in the Immediate scenario once the response to POQ Request is provided (create notification), there will be no further state changes.	Use Case 5: Register for POQ Notifications
DELETE /hub/{{id}}	A request initiated by the Buyer to instruct the Seller to stop sending notifications of POQ create and/or state changes.	Use Case 5: Register for POQ Notifications

Table 2 – Seller Side Endpoints

[R1] The Buyer implementation **MUST** be able to use all REST methods for `/productOfferingQualification` endpoint that are listed in the table above. MEF 79 R1 [9].

6.1.2 Buyer Side Endpoints

Base URL for Cantata:
`https://{{serverBase}}:{{port}}{{?/buyer_prefix}}/mefApi/cantata/productOfferingQualificationNotification/v1/`

Base URL for Sonata:
`https://{{serverBase}}:{{port}}{{?/buyer_prefix}}/mefApi/sonata/productOfferingQualificationNotification/v7/`

The following API endpoints are implemented by the Buyer and are used by the Seller to send POQ related notifications. The endpoints and corresponding data model are defined in `productApi/serviceability/offeringQualification/productOfferingQualificationNotification.api.yaml`.

API Endpoint	Description	MEF 79 Use Case Mapping
POST /listener/poqCreateEvent	A request initiated by the Seller to notify Buyer on POQ creation.	Use Case 9: Notify of POQ State Change
POST /listener/poqStateChangeEvent	A request initiated by the Seller to notify Buyer on POQ state change.	Use Case 9: Notify of POQ State Change
POST /listener/poqItemStateChangeEvent	A request initiated by the Buyer to retrieve full details of a single Product Offering Qualification based on a POQ identifier.	Use Case 9: Notify of POQ State Change

Table 3 – Buyer Side Endpoints

6.1.3 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 [9] Section 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 `productOfferingQualificationManagement.api.yaml` and as attributes of an event (`PoqEvent`) as described in `productOfferingQualificationNotification.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. MEF 79 R80 [9].

[R3] If the Requesting Entity represents precisely one Buyer with the Responding Entity, the request **MUST NOT** specify the `buyerId`. MEF 79 R81 [9].

[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. MEF 79 R82 [9].

[R5] If the Responding Entity represents precisely one Seller to this Buyer, the request **MUST NOT** specify the `sellerId`. MEF 79 R83 [9].

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

6.2 Data Model – Key Entities

Sections below describe the most important entities (aka data types) from the data model which can be found in the API specification (definitions section). Each entity is a simple or composed type (using `allOf` keyword for data types composition). A simple type defines a set of properties that might be of an object, primitive, or reference type.

[R7] If an entity is used in the request or response payload all properties marked as required **MUST** be provided.

A detailed description of the data types is provided in Section 8 and the OpenAPI definition. The examples that illustrate the usage of the data model are included in Section 7.

6.2.1 Request View on Key Entities

Figure 4 depicts a view on the data model that is used in the Product Offering Qualification request (`POST /productOfferingQualification`) that is sent by a Buyer (see Section 6.1.1 for details).

`ProductOfferingQualification_Create` is the root entity of a product offering qualification request. It contains one or more `ProductOfferingQualificationItem_Create`. An item (`ProductOfferingQualificationItem_Create`) defines an item inquiry details (in `MEFProductRefOrValue` structure) and allows for the definition of related contact information (`relatedContactInformation`) or relations to other items (`QualificationItemRelationship`). `MEFProductRefOrValue` allows for the introduction of MEF product-specific properties to the POQ payload. The extension mechanism is described in detail in Section 6.2.3. Also, a `MEFProductRefOrValue` may be used to specify relations to places (using specializations of `RelatePlaceOrValue`) or/and to a product that exists in the Seller's inventory (using `ProductRelationshipWithGrouping`).

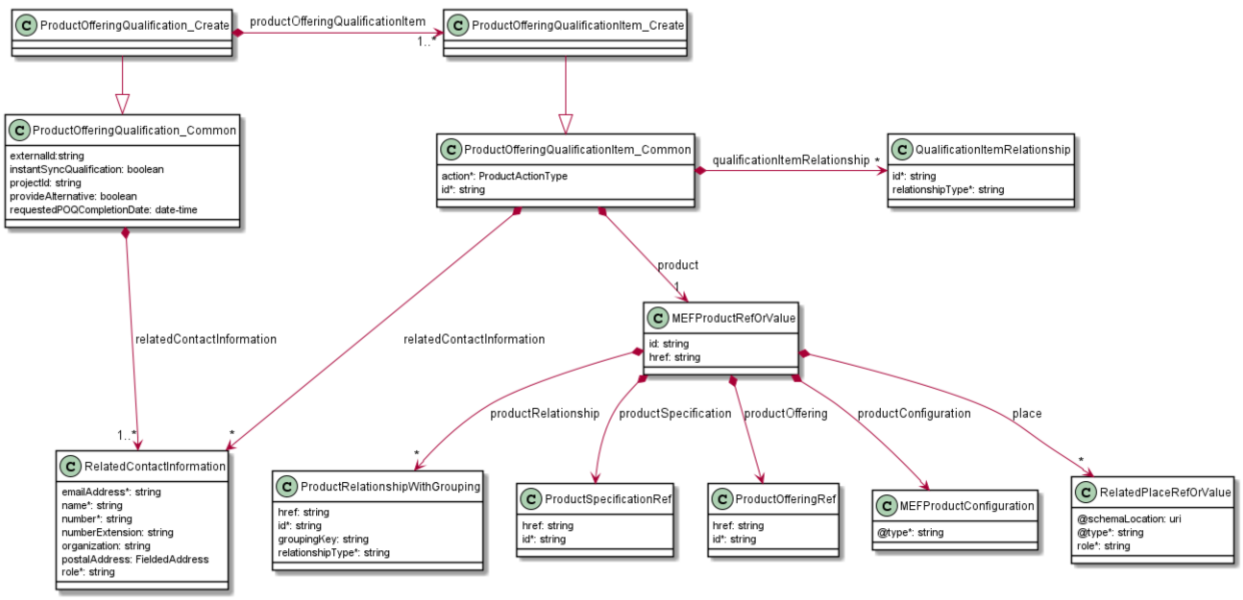
ProductOfferingQualification_Create
productOfferingQualificationItem
1..*
ProductOfferingQualificationItem_Create
ProductOfferingQualification_Common
externalId:string
instantSyncQualification: boolean
projectId: string
provideAlternative: boolean
requestedPOQCompletionDate: date-time
ProductOfferingQualificationItem_Common
action*: ProductActionType
id*: string
qualificationItemRelationship *
QualificationItemRelationship
id*: string
relationshipType*: string
product
1
MEFProductRefOrValue
id: string
href: string
relatedContactInformation
relatedContactInformation
1..*
*
RelatedContactInformation
emailAddress*: string
name*: string
number*: string
numberExtension: string
organization: string
postalAddress: FieldedAddress
role*: string
productRelationship
*
ProductRelationshipWithGrouping
href: string
id*: string
groupingKey: string
relationshipType*: string
productSpecification
ProductSpecificationRef
href: string
id*: string
productOffering
ProductOfferingRef
href: string
id*: string
productConfiguration
MEFProductConfiguration
@type*: string
place
*
RelatedPlaceRefOrValue
@schemaLocation: uri
@type*: string
role*: string

Figure 4 – POQ Management API Data Model – Request View on Key Entities

6.2.2 Response/Inventory View on Key Entities

Figure 5 depicts a view on the data model that is used to provide a response to a Buyer's Product Offering Qualification request (`POST /productOfferingQualification`) or to retrieve POQ by an identifier (`GET /productOfferingQualification/{{id}}`).

`ProductOfferingQualification` is the root entity of a product offering qualification response and it is managed by the Seller. `ProductOfferingQualification` extends `ProductOfferingQualification_Common` (which represents Buyer's request) with a number of attributes, i.e. unique identifier or state information.

[R8] The properties provided in the requests by the Buyer **MUST NOT** be modified by the Seller.

`ProductOfferingQualification` contains one or more items (`ProductOfferingQualificationItem`). For each item, the Seller provides the processing state and if applicable the final response to a particular item from Buyer's request.

[R9] In case of a successful response (`status` equals to `done.ready`) the Seller **MUST** provide serviceability confidence level and if applicable might provide an alternate product proposal (`AlternateProductOfferingProposal`).

[R10] If `status=done.ready`, and the `serviceabilityConfidence` is `yellow` or `green`, the Seller **MUST** specify the `installationInterval` in the response. MEF 79 R56 [9].

[O1] If `status=done.ready`, then the Seller **MAY** specify the `guaranteedUntilDate` in the response. MEF 79 O5 [9].

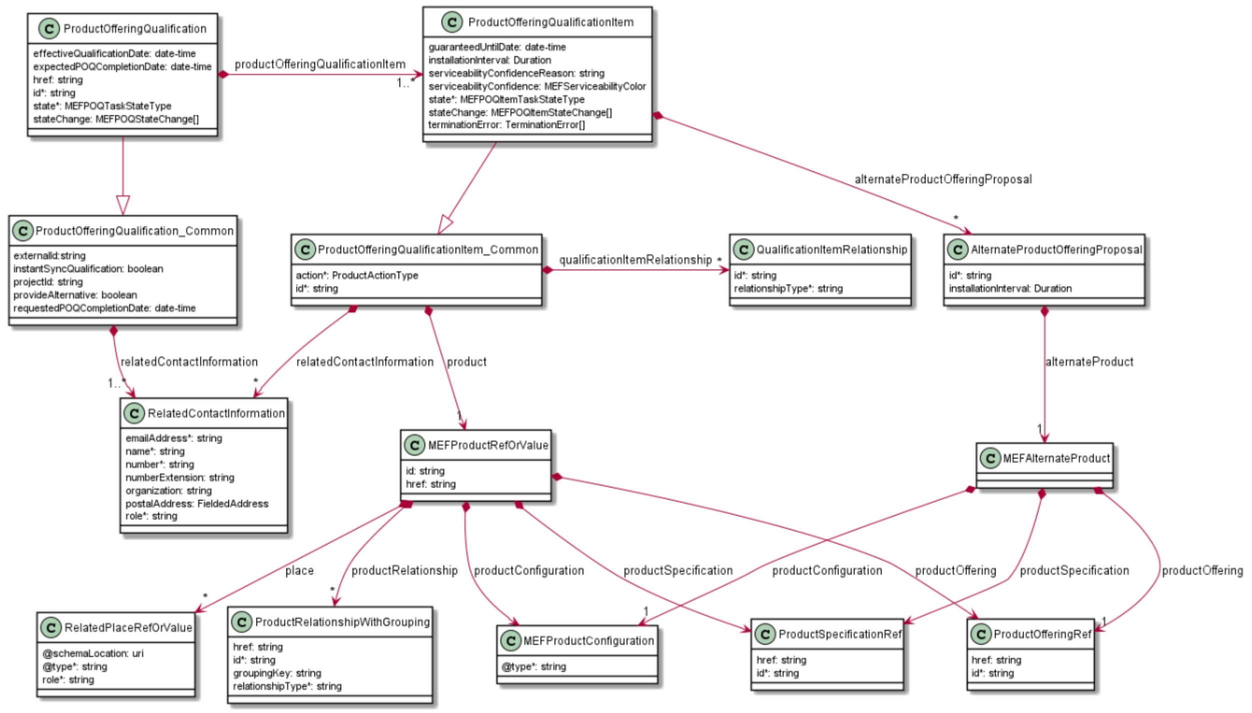
ProductOfferingQualification
effectiveQualificationDate: date-time
expectedPOQCompletionDate: date-time
href: string
id*: string
state*: MEFPOQTaskStateType
stateChange: MEFPOQStateChange[]
productOfferingQualificationItem
1..*
ProductOfferingQualificationItem
guaranteedUntilDate: date-time
installationInterval: Duration
serviceabilityConfidenceReason: string
serviceabilityConfidence: MEFServiceabilityColor
state*: MEFPOQItemTaskStateType
stateChange: MEFPOQItemStateChange[]
terminationError: TerminationError[]
alternateProductOfferingProposal
*
ProductOfferingQualification_Common
externalId:string
instantSyncQualification: boolean
projectId: string
provideAlternative: boolean
requestedPOQCompletionDate: date-time
ProductOfferingQualificationItem_Common
action*: ProductActionType
id*: string
qualificationItemRelationship *
QualificationItemRelationship
id*: string
relationshipType*: string
AlternateProductOfferingProposal
id*: string
installationInterval: Duration
relatedContactInformation
1..*
relatedContactInformation
*
product
1
alternateProduct
1
RelatedContactInformation
emailAddress*: string
name*: string
number*: string
numberExtension: string
organization: string
postalAddress: FieldedAddress
role*: string
MEFProductRefOrValue
id: string
href: string
MEFAlternateProduct
place
*
productRelationship
*
productConfiguration
productSpecification
productConfiguration
1
productOffering
productSpecification
productOffering
1
RelatedPlaceRefOrValue
@schemaLocation: uri
@type*: string
role*: string
ProductRelationshipWithGrouping
href: string
id*: string
groupingKey: string
relationshipType*: string
MEFProductConfiguration
@type*: string
ProductSpecificationRef
href: string
id*: string
ProductOfferingRef
href: string
id*: string

Figure 5 – POQ Management API Data Model – Response/Inventory View on Key Entities

6.2.3 Integration of Product Specifications into Product Offering Qualification Management API

Product specifications are defined using JSON Schema (draft 7) [1] format and are integrated into a POQ payload using TMF extension pattern.

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

The example below shows a header of a Product Specification schema, where `"$id":urn:mef:lso:spec:sonata:access-eline:v1.0.0:poq` is the abovementioned URN:

```
'$schema': http://json-schema.org/draft-07/schema#
'$id': urn:mef:lso:spec:sonata:access-eline:v1.0.0:poq
title: MEF LSO Sonata - Access Eline OVC (POQ) Product Specification
```

Product specifications are provided as JSON schemas without the `MEFProductConfiguration` context.

Product-specific attributes can be introduced into `MEFProductRefOrValue` (defined by the Buyer) or into `AlternateProductOfferingProposal` (may be defined by the Seller while responding to POQ) using `MEFAlternateProduct`. Each of these types introduces the `productConfiguration` attribute of type `MEFProductConfiguration` which is used as an extension point for product-specific attributes.

Implementations might choose to integrate selected product specifications to data model during development. In such case an integrated data model is built and product specifications are in an inheritance relationship with `MEFProductConfiguration` as described in OAS specification [13]. This pattern is called **Static Binding**. The SDK is additionally shipped with a set of API definitions that statically bind all product-related APIs (POQ, Quote, Order, Inventory) with all corresponding product specifications available in the release. The snippets below present an example of a static binding of the POQ API with a number of MEF product specifications, from both `MEFProductConfiguration` and product specification point of view:

```
MEFProductConfiguration:
  description:
    MEFProductConfiguration is used as an extension point for MEF specific
    product/service payload. The `@type` attribute is used as a discriminator
  discriminator:
    mapping:
      urn:mef:lso:spec:sonata:AccessElineOvc:v1.0.0:poq:
'#/components/schemas/AccessElineOvcPoq_v1.0.0'
      urn:mef:lso:spec:cantata-sonata:SubscriberUni:v1.0.0:poq:
'#/components/schemas/SubscriberUniPoq_v1.0.0'
      urn:mef:lso:spec:cantata-sonata:EplEvc:v1.0.0:poq:
'#/components/schemas/EplEvcPoq_v1.0.0'
```

```
      urn:mef:lso:spec:sonata:OperatorUNI:v1.0.0:poq:
'#/components/schemas/OperatorUNIPoq_v1.0.0'
    propertyName: '@type'
  properties:
    '@type':
      description:
        The name of the type, defined in the JSON schema specified above, for
        the product that is the subject of the POQ Request. The named type must
        be a subclass of MEFProductConfiguration.
      type: string
AccessElineOvcPoq_v1.0.0:
  allOf:
    - $ref: '#/components/schemas/MEFProductConfiguration'
    - description:
        OVC Service Attributes control the behavior observable at and between
        External Interfaces to the Carrier Ethernet Network (CEN). The
        behaviors are achieved by the Network Operator and the Operator's
        client (the Service Provider in this case) agreeing on the value for
        each of the Service Attributes.
```

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

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

[R11] `MEFProductConfiguration` type is an extension point that **MUST** be used to integrate product specifications' properties into a request/response payload.

[R12] The `@type` property of `MEFProductConfiguration` **MUST** be used to specify the type of the extending entity.

[R13] Product attributes specified in payload must conform to the product specification indicated by the `@type` property.

Figure 6 below depicts two MEF `<<ProductSpecifications>>` that represent Access E-Line and Operator UNI products. When these products are used in POQ payload the `@type` of `MEFProduct Configuration` takes `"urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq"` or `"urn:mef:lso:spec: sonata:OperatorUNI:1.0.0:poq"` value to indicate which product specification should be used to interpret a set of product-specific attributes included in the payload.

The *poq* suffix after the product type name in the URN comes from the approach that the product schemas may differ depending on the Interface Reference Point function they are used with.

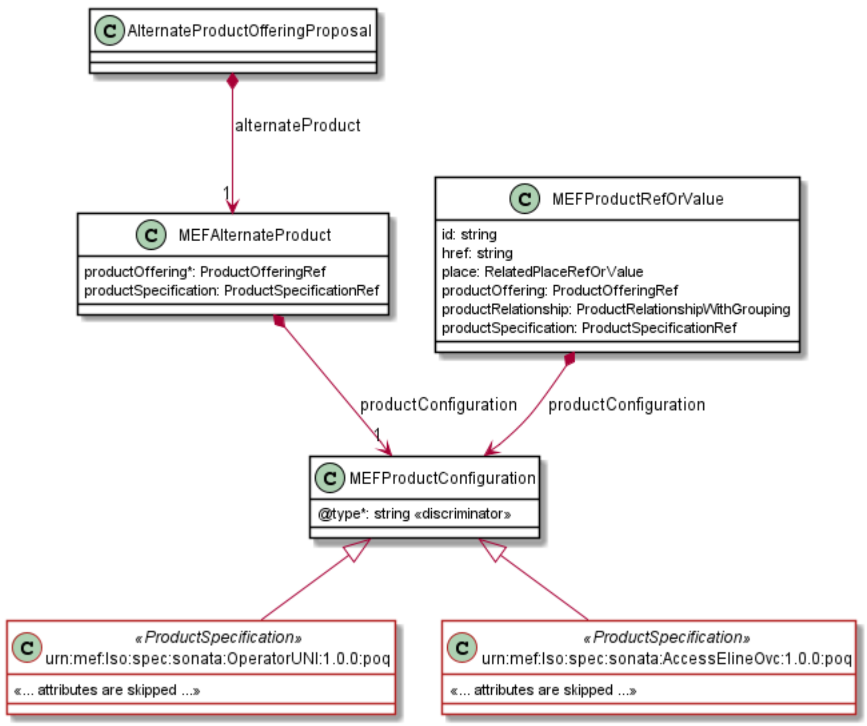
AlternateProductOfferingProposal
alternateProduct
1
MEFAlternateProduct
productOffering*: ProductOfferingRef
productSpecification: ProductSpecificationRef
MEFProductRefOrValue
id: string
href: string
place: RelatedPlaceRefOrValue
productOffering: ProductOfferingRef
productRelationship: ProductRelationshipWithGrouping
productSpecification: ProductSpecificationRef
productConfiguration
1
productConfiguration
MEFProductConfiguration
@type*: string «discriminator»
«ProductSpecification»
urn:mef:lso:spec:sonata:OperatorUNI:1.0.0:poq
«... attributes are skipped ...»
«ProductSpecification»
urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq
«... attributes are skipped ...»

Figure 6 – The Extension Pattern with Sample Product Specific Extension

6.3 Model Structural Validation

The structure of the HTTP payloads exchanged via POQ API endpoints is defined using:

- OpenAPI version 3.0 [13] for product-agnostic part of the payload
- JSON Schema (draft 7) [1] for product-specific part of the payload

[R14] Implementations **MUST** use payloads that conform to these definitions.

[R15] A product specification **MAY** define additional consistency rules and requirements that **MUST** be respected by implementations. These are defined for:

- relations to other items in the same product offering qualification request (e.g. required relation type, multiplicity) (as specified in MEF 79 R37, MEF79 A1-R2 [9])
- relations to entities from the inventory managed by the Seller (as specified in MEF79 R37, MEF79 A1-R1 [9])
- related contact information that is to be defined at an item level
- relations to places (locations) that are to be defined at an item level MEF79 R35 & R36 [9]

6.4 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 being worked on by a separate MEF Project (MEF 128).

7 API Interaction & Flows

This section discusses the most important aspects of end-to-end interactions that result in completed product qualification inquiries. It starts with a description of product specifications which are used in the remainder of the section in example payloads. Then the end-to-end flows are presented for the immediate and deferred interaction patterns. Next, the structure of the POQ request and response is described. This part highlights different variants of POQ interactions for different item action types, place definitions, and alternative responses. Finally, the mechanism of notifications is briefly discussed.

7.1 Sample Product Specification

The Sonata SDK contains draft product specification definitions, from which UNI and Access E-Line are used in the payload samples in this section. In Celine release they are located in the SDK package at:

```
\productSchema\carrierEthernet\accessEline\poq\accessElineOvc.yaml
\productSchema\carrierEthernet\carrierEthernetOperatorUni\poq\carrierEthernetOperatorUni.yaml
```

The product specification data model definitions are available as JSON Schema (draft 7) [1] documents. Figure 7 and Figure 8 depict simplified UML views on these data models in which:

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

The red color in Figure 7 and Figure 8 below highlights the data model of Access E-Line.

Product specifications define a number of product-related and envelope-related requirements. Sample envelope-related requirements for Access E-Line:

- for an Access E-Line product two mandatory relationship roles must be specified, one with the operator ENNI (`ENNI_REFERENCE`) and a second with the operator UNI (`UNI_REFERENCE`) for an `add` action. First must be realized as a product relationship (relation to product existing in Seller's Inventory), second might be realized as a poq item (being part of the same poq) or as a product relationship.

- in the case of a `modify` action, product relationships must have the same value as in the `add` action. They must not be changed.
- for an operator UNI product a place relationship (`INSTALL_LOCATION`) must be specified.
- in the case of a `modify` action, place relationships must have the same value as in the `add` action. They must not be changed.

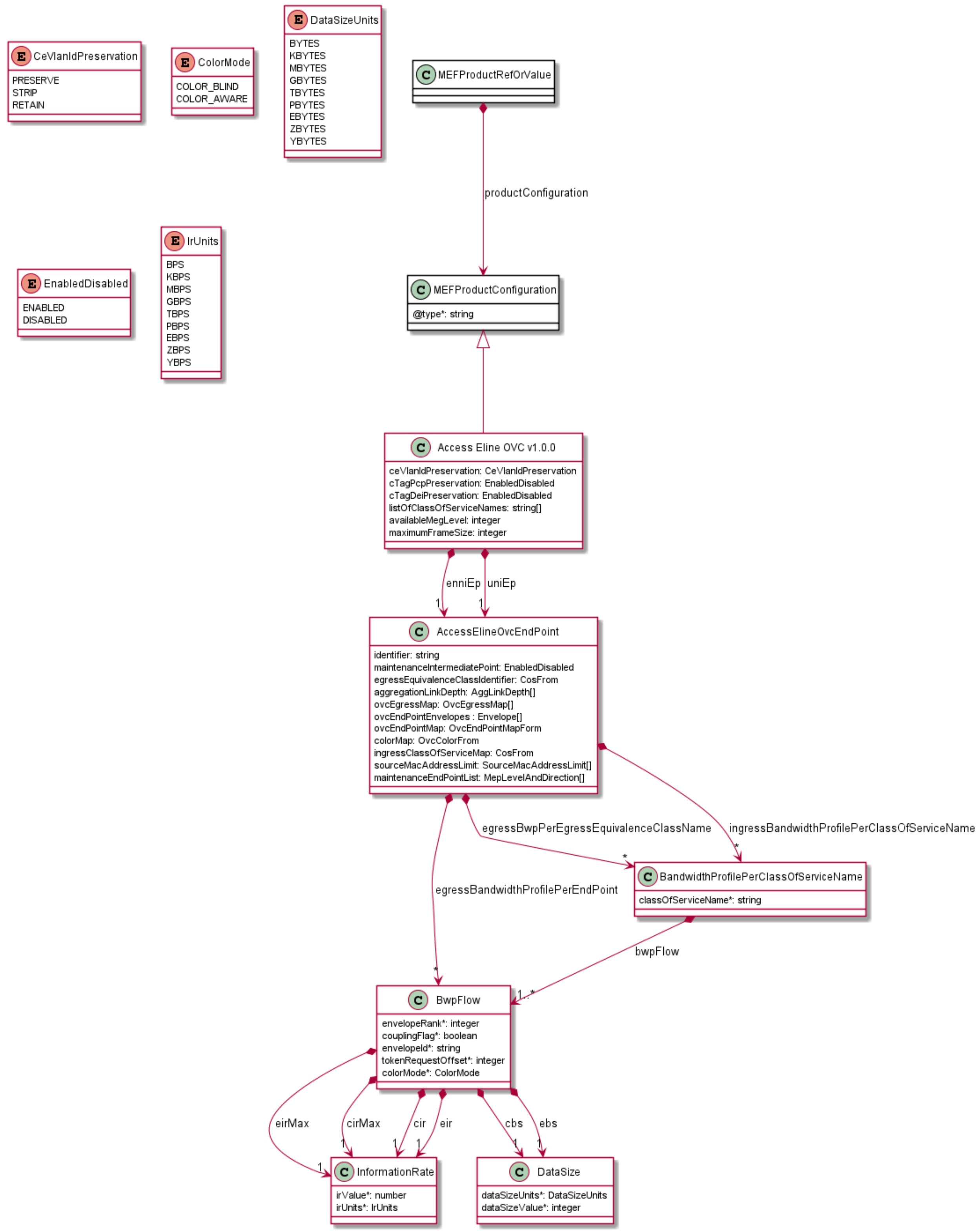
CeVlanIdPreservation
PRESERVE
STRIP
RETAIN
ColorMode
COLOR_BLIND
COLOR_AWARE
DataSizeUnits
BYTES
KBYTES
MBYTES
GBYTES
TBYTES
PBYTES
EBYTES
ZBYTES
YBYTES
MEFProductRefOrValue
productConfiguration
EnabledDisabled
ENABLED
DISABLED
IrUnits
BPS
KBPS
MBPS
GBPS
TBPS
PBPS
EBPS
ZBPS
YBPS
MEFProductConfiguration
@type*: string
Access Eline OVC v1.0.0
ceVlanIdPreservation: CeVlanIdPreservation
cTagPcpPreservation: EnabledDisabled
cTagDeiPreservation: EnabledDisabled
listOfClassOfServiceNames: string[]
availableMegLevel: integer
maximumFrameSize: integer
enniEp
uniEp
1
1
AccessElineOvcEndPoint
identifier: string
maintenanceIntermediatePoint: EnabledDisabled
egressEquivalenceClassIdentifier: CosFrom
aggregationLinkDepth: AggLinkDepth[]
ovcEgressMap: OvcEgressMap[]
ovcEndPointEnvelopes : Envelope[]
ovcEndPointMap: OvcEndPointMapForm
colorMap: OvcColorFrom
ingressClassOfServiceMap: CosFrom
sourceMacAddressLimit: SourceMacAddressLimit[]
maintenanceEndPointList: MepLevelAndDirection[]
egressBwpPerEgressEquivalenceClassName
ingressBandwidthProfilePerClassOfServiceName
*
*
BandwidthProfilePerClassOfServiceName
classOfServiceName*: string
egressBandwidthProfilePerEndPoint
bwpFlow
*
1..*
BwpFlow
envelopeRank*: integer
couplingFlag*: boolean
envelopeId*: string
tokenRequestOffset*: integer
colorMode*: ColorMode
eirMax
cirMax
cir
eir
cbs
ebs
1
1
1
1
1
1
InformationRate
irValue*: number
irUnits*: IrUnits
DataSize
dataSizeUnits*: DataSizeUnits
dataSizeValue*: integer

Figure 7 – A Simplified View on Access E-Line Product Specification Data Model

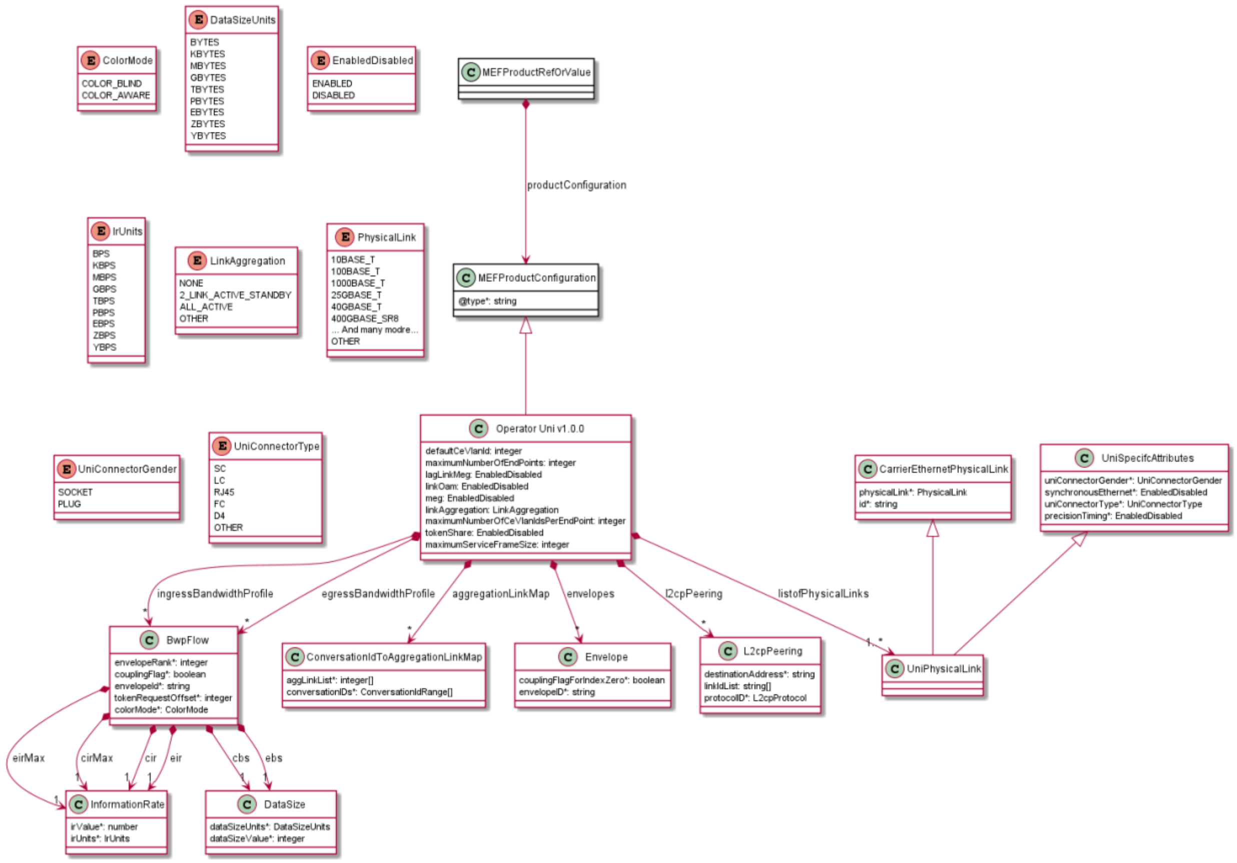

Figure 8 – A Simplified View on UNI Product Specification Data Model

The product relationship (`product.productRelationship`) and the place relationship (`product.place`) are presented in Figure 4 and in more details at Figure 13.

In case some of either product-related or envelope-related requirements are violated the Seller returns an error response to the Buyer which indicates specific functional errors. These errors are listed in the response body (a list of `Error422` entries) for HTTP `422` response.

7.2 Interaction Patterns

To complete the POQ inquiry three interaction patterns can be used depending on Buyer/Seller side capabilities.

7.2.1 Immediate Response

Immediate response can be requested by a Buyer using `instantSyncQualification` flag set to `true`. In case of successful processing, the Seller will respond with POQ in state `done.ready` (indicating success) or `terminatedWithError` (indicating that Buyer has not provided enough information). Otherwise, the appropriate error code and description are returned in case the payload doesn't pass initial validation. Please note that the `done.unableToProvide` state is not supported in the immediate response case. The `done.unableToProvide` state is only used during the Deferred Response pattern when the Seller is unable to provide a response in the time frame required by the Buyer.

Note: The term "Seller Response Code" used in the Business Requirements maps to HTTP response code, where `2xx` indicates *Success* and `4xx` or `5xx` indicate *Failure*.

[R16] The Seller **MAY** provide an immediate answer even when `instantSyncQualification` flag set to `false`. MEF 79 R3 [9].

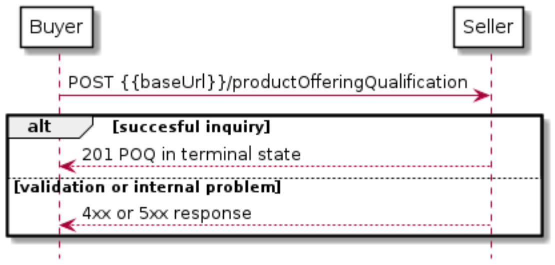
Buyer
Seller
POST {{baseUrl}}/productOfferingQualification
alt
[succesful inquiry]
201 POQ in terminal state
[validation or internal problem]
4xx or 5xx response

Figure 9 – Immediate Response

7.2.2 Deferred Response with Polling

A deferred response can be requested by a Buyer using `instantSyncQualification` flag set to `false`. The Seller responds with partial `POQ` (including at least `id` and `state=acknowledged`) and starts processing the request asynchronously. The Buyer polls the POQ until the final status is reached by the POQ using the POQ identifier specified by the Seller in response.

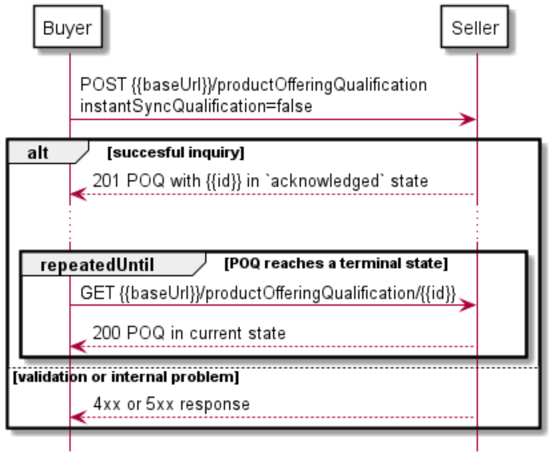
Buyer
Seller
POST {{baseUrl}}/productOfferingQualification
instantSyncQualification=false
alt
[succesful inquiry]
201 POQ with {{id}} in `acknowledged` state
repeatedUntil
[POQ reaches a terminal state]
GET {{baseUrl}}/productOfferingQualification/{{id}}
200 POQ in current state
[validation or internal problem]
4xx or 5xx response

Figure 10 – Deferred Response with Polling

7.2.3 Deferred Response with Notifications

In this variant of the deferred response, the notifications mechanism is used. First, the Buyer registers for notification providing a callback endpoint. Then the Buyer requests for qualification. The Seller sends notifications on POQ creation and subsequent POQ and POQ Item State changes until the final POQ status is reached.

When the Buyer registers for POQ notifications this registration will be valid until the Buyer unsubscribes from the POQ notifications. This implies that for any POQ request the Buyer sends to the Seller, and for which they request a Deferred Response, in the time frame between the registration for the POQ notifications and unsubscribing from the same POQ notifications by the Buyer, the Seller will have to inform the Buyer of changes using the POQ notifications.

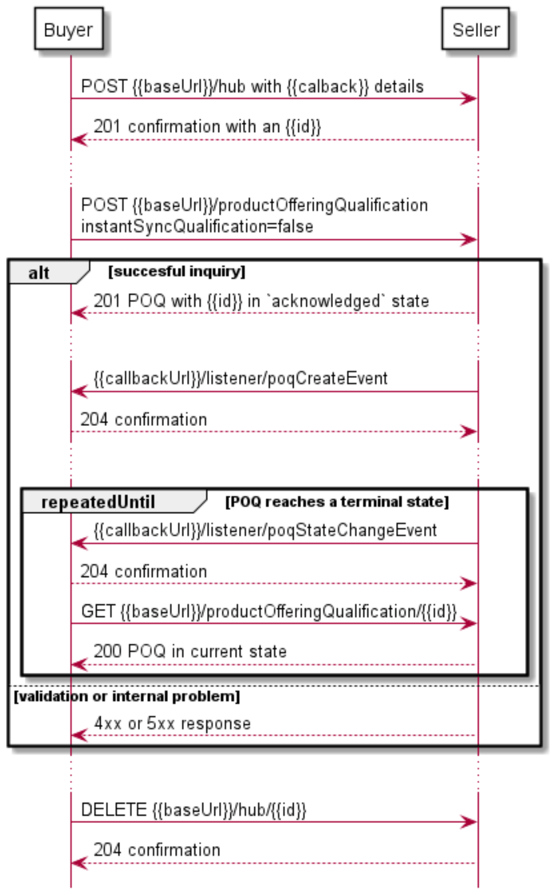
Buyer
Seller
POST {{baseUrl}}/hub with {{calback}} details
201 confirmation with an {{id}}
POST {{baseUrl}}/productOfferingQualification instantSyncQualification=false
alt
[succesful inquiry]
201 POQ with {{id}} in `acknowledged` state
{{callbackUrl}}/listener/poqCreateEvent
204 confirmation
repeatedUntil
[POQ reaches a terminal state]
{{callbackUrl}}/listener/poqStateChangeEvent
204 confirmation
GET {{baseUrl}}/productOfferingQualification/{{id}}
200 POQ in current state
[validation or internal problem]
4xx or 5xx response
DELETE {{baseUrl}}/hub/{{id}}
204 confirmation

Figure 11 – Deferred Response with Notifications

7.3 Sending Product Offering Qualification Request

To send a POQ request a Buyer must use createProductOfferingQualification (`POST {{baseUrl}}/productOfferingQualification`) which represents Use Case 6 from MEF 79 [9]. In the remainder of this section, some of the POQ payloads attributes might be omitted to simplify examples' content. The full list of attributes is available in Section 8 and in the API specification which is an integral part of this standard.

7.3.1 Buyer POQ Request

The most common Buyer attributes used in a request are listed below:

```
{
  "provideAlternative" : false,
  "requestedPOQCompletionDate" : "2020-08-29T07:03:36.2640125Z",
  "instantSyncQualification" : false,
  "projectId" : "ourProjectForCustomerX",
  "relatedContactInformation" : [],
  "productOfferingQualificationItem" : [ {
    "id" : "item-001",
    "action" : "add",
    "product": { << product specific attributes and configuration >> },
    "relatedContactInformation": [ << required related contact information entries >> ],
    "qualificationItemRelationship": [ << item level relationships >> ]
  },
  {
      "id": "item-002"
      <<rest of item-002 attributes are omitted>>
  }
  ]
}
```

[R17] Each POQ request sent by the Buyer **MUST** include at least one POQ Item. MEF 79 R28 [9].

[R18] The Buyer **MUST** set the `instantSyncQualification` flag to request for an immediate or deferred response. MEF 79 R27 [9].

[R19] The `requestedPOQCompletionDate` **MUST** be specified when `instantSyncQualification=false`.

[R20] The Related Contact Information list **MUST** contain an entry that represents the Buyer Contact Information. MEF 79 R27 [9].

[R21] When specifying the Buyer Contact Information the Buyer **MUST** provide a `relatedContactInformation` entry with a `role=buyerContactInformation`.

Note: During onboarding the Seller may require to provide an additional contact `role`.

Note: It is up to Seller's discretion on how to react in case the Buyer provides a contact `role` that is not listed by this standard or agreed upon during the onboarding. Preferably the Seller should return an error with a message stating which `roles` are accepted. It may also be ignored.

[R22] The Buyer **MUST** follow product specification related requirements when specifying values for `productOfferingQualificationItem.relatedContactInformation`.

Note: MEF 79 R27 [9] specifies the attributes that must be provided by a Buyer in the POQ request. The API model is not compatible with this requirement in following way:

- `provideAlternative` defines a default `false` value in case it is not provided by the Buyer,

- `instantSyncQualification` defines a default `false` value in case it is not provided by the Buyer.

The Buyer can request alternative proposals (`provideAlternative=true`) in the Create POQ request. In case the Seller cannot support the requested Product Offering or Product Configuration, alternative Product Offerings which the Seller does support can be included in the Create POQ response by the Seller.

7.3.2 Seller's Response to POQ Request

The Seller deferred response may look like this:

```
{
  "provideAlternative" : false,
  "requestedPOQCompletionDate" : "2020-08-29T07:03:36.264012Z",
  "instantSyncQualification" : false,
  "projectId" : "ourProjectForCustomerX",
  "relatedContactInformation" : [],
  "productOfferingQualificationItem" : [ {
    "id" : "item-001",
    "action" : "add",
    "product": { << product specific attributes and configuration >> },
    "relatedContactInformation": [ << required related contact information entries >> ],
    "qualificationItemRelationship": [ << item level relationships >> ]
  },
  {
      "id": "item-002"
      <<rest of item-002 attributes are omitted>>
  }
  ]
}
```

[R23] The Seller **MUST NOT** change the values of attributes specified by the Buyer.

These attributes are indicated above with an appropriate comment. The Seller might append related contact information, if required, either at item or POQ level but cannot modify related contact information provided by the Buyer.

[R24] If the `ProductOfferingQualification` is used to provide a response to an instant qualification request the `state` **MUST** take either `terminatedWithError` or `done.ready` value. MEF 79 R45 [9].

[R25] The Related Contact Information list **MUST** contain an entry that represents Seller Contact Information. MEF 79 R48 & R72 [9].

[R26] When specifying the Seller Contact Information the Seller **MUST** provide a `relatedContactInformation` entry with a `role=sellerContactInformation`.

[R27] Each item in `productOfferingQualificationItem` list in the response **MUST** correspond to an item from a list in the request. MEF 79 R50 & R74 [9].

[R28] When the item state is `done.ready` the Seller **MUST** provide value for the `serviceabilityConfidence` MEF 79 R55 & R78 [9] attribute.

[R29] When the attribute `serviceabilityConfidence` is set to `green` or `yellow` the Seller **MUST** provide values for the `installationInterval` attribute. MEF 79 R56 & R79 [9].

[R30] When the item state is `terminatedWithError` or `done.abandoned` the Seller **MUST NOT** provide values for the `serviceabilityConfidence`, `installationInterval`, `validFor`, and `alternateProductOfferingProposal` attributes. MEF 79 R53 & R76 [9].

[R31] When the item state is `terminatedWithError` the Seller **MUST** provide value for `terminationError` attribute. MEF 79 R54 & R77 [9].

7.3.3 POQ Item Specification Details

[R32] For each item, the Buyer **MUST** specify `id`, `action`, and `product`.

In the remainder of this section examples of POQ item definitions for different `action` types are given.

7.3.3.1 POQ Item Structure for `add` Action

The example below represents a single POQ request item (`ProductOfferingRequestItem_Create`) to evaluate the installation of a new (action `add`) Access E-Line product (type `urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq`). A Buyer is sending all mandatory attributes as well as a subset of optional attributes that are relevant for this inquiry.

```
{
  "action": "add",
  "id": "item-002",
  "product": {
    "productOffering": {
      "id": "000073"
    },
    "productConfiguration": {
      "@type": "urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq",
      "enniEp": {
        "ingressBandwidthProfilePerClassOfServiceName": [
          {
            "classOfServiceName": "silver",
            "bwpFlow": [
              {
                "envelopeRank": 1,
                "couplingFlag": false,
                "envelopeName": "defaultENNI",
                "tokenRequestedOffset": 0,
                "colorMode": "COLOR_BLIND",
                "cir": {
                  "irValue": 20,
                  "irUnits": "MBPS"
                },
                "cbs": {
                  "dataSizeValue": 50,
                  "dataSizeUnits": "KBYTES"
                },
                "eir": {
                  "irValue": 0,
                  "irUnits": "BPS"
                },
                "ebs": {
                  "dataSizeValue": 0,
                  "dataSizeUnits": "BYTES"
                },
                "cirMax": {
                  "irValue": 20,
                  "irUnits": "MBPS"
                },
                "eirMax": {
                  "irValue": 0,
                  "irUnits": "BPS"
                }
              }
            ]
          }
        ]
      },
      "maximumFrameSize": 1522,
      "uniEp": {
        "ingressBandwidthProfilePerClassOfServiceName": [
          {
            "classOfServiceName": "silver",
            "bwpFlow": [
              {
                "envelopeRank": 1,
                "couplingFlag": false,
                "envelopeName": "defaultUNI",
                "tokenRequestedOffset": 0,
                "colorMode": "COLOR_BLIND",
                "cir": {
                  "irValue": 20,
                  "irUnits": "MBPS"
                },
                "cbs": {
                  "dataSizeValue": 50,
                  "dataSizeUnits": "KBYTES"
                },
                "eir": {
                  "irValue": 0,
                  "irUnits": "BPS"
                },
                "ebs": {
                  "dataSizeValue": 0,
                  "dataSizeUnits": "BYTES"
                },
                "cirMax": {
                  "irValue": 20,
                  "irUnits": "MBPS"
                },
                "eirMax": {
                  "irValue": 0,
                  "irUnits": "BPS"
                }
              }
            ]
          }
        ]
      }
    },
    "productRelationship": [
      {
        "relationshipType": "ENNI_REFERENCE",
        "id": "00000000-0000-000a-0000-000000000001"
      }
    ]
  },
  "qualificationItemRelationship": [
    {
      "relationshipType": "UNI_REFERENCE",
      "id": "item-001"
    }
  ],
  "relatedContactInformation": [
    {
      "number": "1-430-358-5262",
      "emailAddress": "john@mef.net",
      "role": "technical team",
      "name": "John Doe"
    }
  ]
}
```

MEF 79 defines the `productOffering` as mandatory for the `add` action. This is in the case the Buyer knows exactly what Product Offering to refer to in the request. However, the API allows additionally the use case of providing the `productSpecification` instead. In this scenario, the Buyer asks if a particular type of product (defined by providing `productSpecification` and additional product-specific attributes in `productConfiguration`) can be served and gets back from the Seller a direct or alternate `productOfferings` in the response. These can be later used in Quote and Product Order steps.

[R33] If `action=add` the Buyer **MUST** provide exactly one of `productOffering` or `productSpecification`. MEF 79 R31 [9].

[R34] If `action=add` the Buyer **MUST** provide product-specific attributes (`productConfiguration`). MEF 79 R31 [9].

[R35] The product specified in `product.@type` **MUST** be compatible with the one defined by `productOffering` or `productSpecification`.

[R36] If `action=add` the Buyer **MUST NOT** provide `product.id`. MEF 79 R30 [9].

In the example above, the Access E-Line product specification is identified as `00000000-0000-0000-0000-0000000007e3`` in the Seller's catalog. This specification describes the structure and requirements defined for this product which should be validated. An Access E-Line product specification defines two mandatory relationship types that have to be specified in case of an `add` action qualification: `ENNI_REFERENCE` and `UNI_REFERENCE`. The reference to an operator UNI product might use another POQ item or an existing product from the Seller's inventory. This example assumes that UNI product is another item of the POQ request with a unique identifier `item-001`. This Access E-Line product references an existing ENNI product that is uniquely identified with id `00000000-0000-000a-0000-000000000001` in the Seller's inventory.

The place is not provided as Access E-Line product specification does not allow for a place description to be part of the request. Values for some of the available product attributes are provided under `productConfiguration` node. This example uses a tiny subset of available Access E-Line attributes.

7.3.3.2 POQ Item Structure for `modify` Action

The example below represents a single POQ request item to evaluate a modification of an existing (action `modify`) Access E-Line product (type `urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq`).

```
{
  "action": "modify",
  "id": "item-001",
  "product": {
    "id": "01494079-6c79-4a25-83f7-48284196d44d",
    "productConfiguration": {
      "@type": "urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq",
      "enniEp": {
        "ingressBandwidthProfilePerClassOfServiceName": [
          {
            "classOfServiceName": "silver",
            "bwpFlow": [
              {
                "envelopeRank": 1,
                "couplingFlag": false,
                "envelopeName": "defaultENNI",
                "tokenRequestedOffset": 0,
                "colorMode": "COLOR_BLIND",
                "cir": {
                  "irValue": 40,
                  "irUnits": "MBPS"
                },
                "cbs": {
                  "dataSizeValue": 100,
                  "dataSizeUnits": "KBYTES"
                },
                "eir": {
                  "irValue": 0,
                  "irUnits": "BPS"
                },
                "ebs": {
                  "dataSizeValue": 0,
                  "dataSizeUnits": "BYTES"
                },
                "cirMax": {
                  "irValue": 40,
                  "irUnits": "MBPS"
                },
                "eirMax": {
                  "irValue": 0,
                  "irUnits": "BPS"
                }
              }
            ]
          }
        ]
      },
      "maximumFrameSize": 1522,
      "uniEp": {
        "ingressBandwidthProfilePerClassOfServiceName": [
          {
            "classOfServiceName": "silver",
```

```json
            "bwpFlow": [
              {
                "envelopeRank": 1,
                "couplingFlag": false,
                "envelopeName": "defaultUNI",
                "tokenRequestedOffset": 0,
                "colorMode": "COLOR_BLIND",
                "cir": {
                  "irValue": 40,
                  "irUnits": "MBPS"
                },
                "cbs": {
                  "dataSizeValue": 100,
                  "dataSizeUnits": "KBYTES"
                },
                "eir": {
                  "irValue": 0,
                  "irUnits": "BPS"
                },
                "ebs": {
                  "dataSizeValue": 0,
                  "dataSizeUnits": "BYTES"
                },
                "cirMax": {
                  "irValue": 40,
                  "irUnits": "MBPS"
                },
                "eirMax": {
                  "irValue": 0,
                  "irUnits": "BPS"
                }
              }
            ]
          }
        ]
      }
    }
  },
  "relatedContactInformation": [
    {
      "number": "1-430-358-5262",
      "emailAddress": "john@mef.net",
      "role": "technical team",
      "name": "John Doe"
    }
  ]
}
```

[R37] The modify request **MUST** specify a reference (provide `product.id`) to an existing product which is a subject of this order and provide the desired `product.productConfiguration`. MEF 79 R32 [9].

[R38] The modify request **MUST** repeat the same values (specified or empty) of `product.productOffering`, `product.productRelationship`, `product.productSpecification`, and `product.place` as they are for available in the inventory for a given product instance. These values cannot be updated nor deleted.

There is no possibility to send an update to single attributes. The Buyer must send a full `product.productConfiguration` together with `product.productRelationship` and `product.place` (if set), which means all attributes that represent the desired state, even if some of them do not change. If Seller does not allow for some of the attributes to change an appropriate error response (`422`) must be returned to the Buyer.

The above example represents a qualification request for an Access E-Line product change. In particular, changes to `cir` (Committed Information Rate) values for `ENNI` and `UNI` bandwidth profiles are qualified. The Access E-Line product exists in Seller's inventory and is identified as `01494079-6c79-4a25-83f7-48284196d44d`.

Note: The Buyer can update a Buyer-related contact by providing a full list of existing `relatedContactInformation` items, and updating the value those with Buyer-related `roles`.

Note: The Buyer can not update a Buyer-related note. New notes can only be appended to existing list of `note` items.

7.3.3.3 POQ Item Structure for `delete` Action

The example below represents a single POQ request item to evaluate a disconnect of an existing (action `delete`) Access E-Line product (type `urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq`).

```
{
  "id": "item-001",
  "action": "remove",
  "product": {
    "id": "01494079-6c79-4a25-83f7-48284196d44d"
  }
}
```

Product instance identifier (`01494079-6c79-4a25-83f7-48284196d44d`) is the only allowed attribute in the disconnect evaluation request.

[R39] If `action=delete` the Buyer **MUST NOT** provide `productOffering`, `productSpecification` and `productConfiguration` attributes. MEF 79 R33 [9].

[R40] If the item action is set to `modify` or `delete` the Buyer **MUST** provide the `product.id` of the referenced Product. MEF 79 R32 [9].

[R41] In the `delete` request the only allowed attributes of `product` are `id` and `@type`. MEF 79 R33 [9].

7.3.4 Specifying Place Details

Some product specifications may define requirements concerning place definition in case an `add` or `modify` action is used. For example, the specification of a place in the case of Access E-Line product is not allowed. Operator UNI product specification requires an `INSTALL_LOCATION` place definition in case of an `add` action.

There are different formats in which place information may be provided: geographic point (`MEFGeographicPoint`), fielded (`FieldedAddress`), formatted (`FormattedAddress`), geographic address label (`GeographicAddressLabel`), geographic site reference (`GeographicSiteRef`), and a geographic address reference (`GeographicAddressRef`). Four of them can be used to provide a full place description in POQ or POQ item. The site and address reference allow specifying the place information as a reference to a previously validated address or site available through Seller's Address and Site API endpoints. To distinguish between the place types the `@type` discriminator is used.

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

Note: The *RefOrValue* stands for a pattern where an address can be provided either by `id` (using `GeographicSiteRef` or `GeographicAddressRef`) OR by value (with use of `MEFGeographicPoint`, `FieldedAddress`, `FormattedAddress`, `GeographicAddressLabel`). There is no way to specify an address with use of both ref AND value at the same time.

[R42] Buyer and Seller **MUST** support at least one of `FieldedAddress` or `FormattedAddress` place representations. MEF 79 R84 & R85 [9].

[R43] `GeographicAddressRef` or `GeographicSiteRef` **MUST** be used to provide place information by reference. This method is referred to as the "Known Address ID method" in MEF 79 Section 8.9.3.1. MEF79 D2 [9].

Examples of different place specification formats are provided below.

7.3.4.1 Fielded Address

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

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

7.3.4.2 Formatted Address

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

Place information in a form of a formatted address. The type discriminator has the value `FormattedAddress`. This example contains the same information as `FieldedAddress` example provided above, however, the detailed floor information is omitted. The second address line (`addrLine2`) could be used to specify that information.

7.3.4.3 Geographic Point

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

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

7.3.4.4 Geographic Address Label

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

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

7.3.4.5 Geographic Site Reference

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

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

7.3.4.6 Geographic Address Reference

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

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

7.4 Retrieving POQ Information

7.4.1 Retrieving POQ List

A Buyer can retrieve a list of the POQs by using `GET /productOfferingQualification/` operation with desired filtering criteria.

MEF 79 [9] specifies the possible filtering criteria, in MEF 79 O7:

- `state`
- `projectId`
- `requestedPOQCompletionDate.lt`
- `requestedPOQCompletionDate.gt`

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 the URI query format [3]. Section 8.1.2 provides details about the implementation of the pagination mechanism.

The correct response (HTTP code `200`) contains a list of POQs matching the criteria in the response body. In case there are no POQs matching the criteria an empty list is returned. MEF 79 [9] narrows the list of POQ attributes that are populated in the response to:

- `id`,
- `state`,
- `requestedPOQCompletionDate`,
- `projectId`.

The Buyer may request to retrieve all POQ records that are associated with the project `p1` by executing the following query:

```
https://seller.com/mefApi/sonata/productOfferingQualification/v7/
productOfferingQualification?projectId=p1
```

In the response, the Seller returns all POQ matching these filtering criteria.

```
[
  {
    "requestedPOQCompletionDate": "2020-09-18",
    "id": "97975e56-b6ba-40d4-b9b3-dab2b0e58279",
    "state": "inProgress",
    "projectId": "p1"
  },
  {
    "requestedPOQCompletionDate": "2020-09-26",
    "id": "79de3367-ce55-4e9a-952c-3c16e715bb7f",
    "state": "done.ready",
    "projectId": "p1"
  }
]
```

To see full details of a particular POQ the Buyer can retrieve POQ by the identifier as described in the section below.

7.4.2 Retrieving POQ by Identifier

POQ information can be retrieved from the Seller using `GET /productOfferingQualification/{{id}}` operation. The correct payload returned in the response includes all the attributes Buyer has provided while sending POQ request. In case `id` does not identify a POQ that belongs to the Buyer an error response `404` must be returned. A correct response includes at least POQ unique `id` and up-to-date `state` assigned to POQ and all its items.

POQ can be in an intermediate (`inProgress`) or one of the final states (`done.ready`, `done.unableToProvide`, or `terminatedWithError`). Details about POQ structure and allowed POQ states are provided in Section 8.2.

```
{
  "id" : "00000000-0000-0000-0000-000000000b01",
  "href" : "{{baseUrl}}/productOfferingQualification/00000000-0000-0000-0000-000000000b01",
  "state" : "done.ready",
   << some attributes are omitted >>
  "productOfferingQualificationItem" : [ {
    "id" : "item-001",
    "state" : "done.ready",
    "action" : "add",
    "serviceabilityConfidence" : "green",
    "guaranteedUntilDate" : "2020-12-23T12:00:00+02:00",
    << some attributes are omitted >>
    "stateChange" : [ {
      "changeDate" : "2020-08-23T21:13:32.20+02:00",
      "state" : "acknowledged"
    }, {
      "changeDate" : "2020-08-23T21:13:32.25+02:00",
      "state" : "inProgress"
    }, {
      "changeDate" : "2020-08-23T21:28:49.12+02:00",
      "state" : "done.ready"
    } ]
  } ],
  "stateChange" : [{
      "changeDate" : "2020-08-23T21:13:32.20+02:00",
      "state" : "acknowledged"
    }, {
    "changeDate" : "2020-08-23T21:13:32.25+02:00",
    "state" : "inProgress"
  }, {
    "changeDate" : "2020-08-23T21:28:50.25+02:00",
    "state" : "done.ready"
  } ]
}
```

In the above example, some of the attributes initially specified by the Buyer are removed from the payload to make the example more concise. The POQ is in a `done.ready` final state. This is only possible if all items are in the `done.ready` state as well. The Seller's response to an inquiry represented by id `item-001` is positive (serviceability confidence is `green`) and valid almost to the end of 2020 (`guaranteedUntilDate`). In the state transition history (`stateChange`) all transitions for an item and POQ are listed. As listed in history the processing of the POQ request was almost instantaneous – completion under 20 seconds.

7.5 Alternative Product Proposals

The Buyer may be interested in an alternative response in case the Seller is not able to qualify an exact product. In that case, the Seller may respond with a non-empty list of alternatives for each item with a confidence level assigned to either `yellow` or `red`.

For example, a Buyer is requesting an option for an alternative response:

```
{
  "provideAlternative" : true,
  "requestedPOQCompletionDate" : "2020-08-12T18:45:33.228518Z",
  "instantSyncQualification" : false,
  ...
}
```

As the Seller is not able to deliver the exact product, a single alternative is returned:

```
{
  "id" : "00000000-0000-0000-0000-000000000b01",
  "href" : "{{baseUrl}}/productOfferingQualification/00000000-0000-0000-0000-000000000b01",
  << some attributes are omitted >>
  "productOfferingQualificationItem" : [{
    "action" : "add",
    "id" : "item-001",
    "product" : {
      "productConfiguration" : {
        "@type" : "urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq",
        "uniEp": {
            "ingressBandwidthProfilePerClassOfServiceName": [
                {
                    "classOfServiceName": "silver",
                    "bwpFlow": [
                        {
                            "envelopeRank": 1,
                            "couplingFlag": false,
                            "envelopeName": "defaultUNI",
                            "tokenRequestedOffset": 0,
                            "colorMode": "COLOR_BLIND",
                            "cir": {
                                "irValue": 20,
                                "irUnits": "MBPS"
                            },
                            << some attributes are omitted >>
                        }
                    ]
                }
            ]
        }
        << some attributes are omitted >>
      },
      "productOffering": {
          "id": "000073"
      },
      << some attributes are omitted >>
    },
    "serviceabilityConfidence" : "red",
    "alternateProductOfferingProposal" : [ {
      "installationInterval" : {
        "amount" : 10,
        "timeUnit" : "days"
      },
      "id" : "alternative-01",
      "alternateProduct" : {
        "productOffering": {
          "id": "000099"
        },
        "productSpecification": {
          "id": "urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq"
        },
        "productConfiguration" : {
            "@type" : "urn:mef:lso:spec:sonata:AccessElineOvc:1.0.0:poq",
            "uniEp": {
                "ingressBandwidthProfilePerClassOfServiceName": [
                    {
                        "classOfServiceName": "silver",
                        "bwpFlow": [
                            {
                                "envelopeRank": 1,
                                "couplingFlag": false,
                                "envelopeName": "defaultUNI",
                                "tokenRequestedOffset": 0,
                                "colorMode": "COLOR_BLIND",
                                "cir": {
                                    "irValue": 30,
                                    "irUnits": "MBPS"
                                },
                                << some attributes are omitted >>
                            }
                        ]
                    }
                ]
            },
          << some attributes are omitted >>
        }
      }
    } ],
    << some attributes are omitted >>
    "state" : "done.ready"
  } ],
  << some attributes are omitted >>
}
```

The Seller is not able (`serviceabilityConfidence` is `red`) to deliver an Access E-Line service with `cir` set to 20 Mbps. However, there is an option for a service with `cir` of 30 Mbps. This response is returned as `alternative-01`. The installation interval for the alternative offering is 10 days.

7.6 Notifications

Notifications are used to asynchronously inform the Buyer about:

- the creation of a `ProductOfferingQualification`,
- `ProductOfferingQualification.state` attribute value change,
- `ProductOfferingQualificationItem.state` attribute value change

Notifications are sent from Seller to Buyer in case:

- both Seller and Buyer support notification mechanism
- Buyer has registered to receive notifications from the Seller
- The state change notifications are sent only in the Deferred scenario as in the Immediate scenario once the response to POQ Request is provided (create notification), there will be no further state changes. (Section 7.2.3)

To register for notifications Buyer can either register for all POQ notifications:

```
{
  "callback": "https://buyer.co/listenerEndpoint"
}
```

or be more selective using `query` attribute:

```
{
  "callback": "https://buyer.co/listenerEndpoint",
  "query": "eventType=poqCreateEvent"
}
```

Seller sends notifications about POQ create and update events. Example of POQ create event might look like:

```
{
  "eventId": "event-001",
  "eventType": "poqCreateEvent",
  "eventTime": "2020-08-07T01:07:42.7030052+01:00",
  "event": {
    "id": "00000000-0000-0000-0000-000000000b01"
  }
}
```

Notifications are sent to:

- `https://buyer.co/listenerEndpoint/mefApi/sonata/productOfferingQualificationNotification/v7/listener/poqCreateEvent` in case of `poqCreateEvent`
- `https://buyer.co/listenerEndpoint/mefApi/sonata/productOfferingQualificationNotification/v7/listener/poqStateChangeEvent` in case of `poqStateChangeEvent`
- `https://buyer.co/listenerEndpoint/mefApi/sonata/productOfferingQualificationNotification/v7/listener/poqItemStateChangeEvent` in case of `poqItemStateChangeEvent`

[R44] Seller **MUST** send events only to Buyers who have registered to receive such notifications. MEF 79 R58 [9].

[R45] The create notifications **MUST** be sent in both Immediate and Deferred scenarios. MEF 79 R 59 [9].

[R46] The state change notifications **MUST** be sent only in the Deferred scenario (there will be no state changes in the Immediate scenario anyway). MEF 79 R 59 [9].

Note: MEF 79 defines only one type of the update event: `poqStateChangeEvent` and specifies that both `Poq` and `PoqItem` state changes notifications must be sent using this type. In other MEF APIs it is a common pattern to separate the endpoint for Items to allow the Buyer to choose the level of notifications and avoid unwanted messages. This pattern has also been applied in the POQ API.

8 API Details

8.1 API Patterns

8.1.1 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 POQ API uses the error responses depicted and described below.

Implementations can use HTTP error codes not specified in this standard in compliance with rules defined in RFC 7231 [5]. In such a case the error message body structure might be aligned with the `Error`.

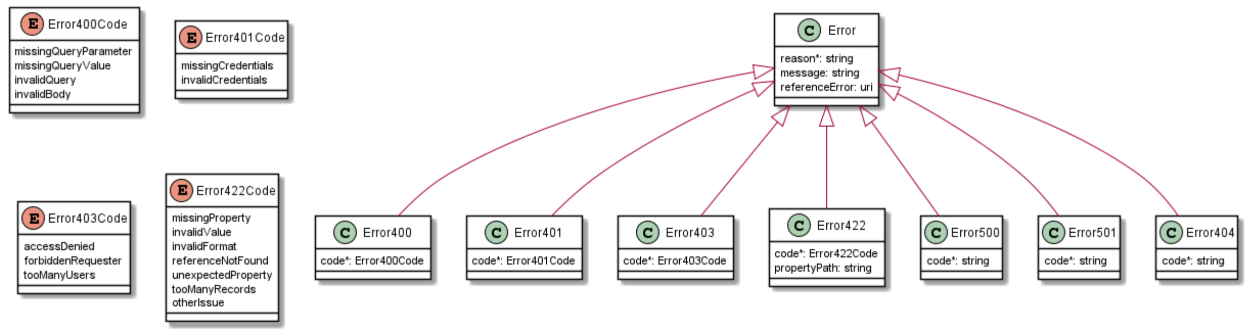

Figure 12 – Data Model Types to Represent an Erroneous Response

8.1.1.1 Type Error

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

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

Table 4 – Type Error

8.1.1.2 Type Error400

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

Inherits from:

- Error

Name	Type	Description
`code*`	string	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.

Table 5 – Type Error400

8.1.1.3 Type Error401

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

Inherits from:

- Error

Name	Type	Description
code*	string	One of the following error codes: - `missingCredentials`: No credentials provided. - `invalidCredentials`: Provided credentials are invalid or expired.

Table 6 – Type Error401

8.1.1.4 Type Error403

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

Inherits from:

- Error

Name	Type	Description
code*	string	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.

Table 7 – Type Error403

8.1.1.5 Type Error404

Description: Resource for the requested path not found. (https://datatracker.ietf.org/doc/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.

Table 8 – Type Error404

8.1.1.6 Type Error422

The response for HTTP status `422` is a list of elements that are structured using the `Error422` data type. Each list item describes a business validation problem. This type introduces the `propertyPath` attribute which points to the erroneous property of the request, so that the Buyer may fix it easier.

It is highly recommended that this property should be used, yet remains optional because it might be hard to implement.

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

Inherits from:

- Error

Name	Type	Description
`code*`	string	One of the following error codes: - `missingProperty`: The property the Seller has expected is not present in the payload. - `invalidValue`: The property has an incorrect value. - `invalidFormat`: The property value does not comply with the expected value format. - `referenceNotFound`: The object referenced by the property cannot be identified in the Seller system. - `unexpectedProperty`: Additional property, not expected by the Seller has been provided. - `tooManyRecords`: the number of records to be provided in the response exceeds the Seller's threshold. - `otherIssue`: Other problem was identified (detailed information provided in a reason).
`propertyPath`	string	A pointer to a particular property of the payload that caused the validation issue. It is highly recommended that this property should be used. Defined using JavaScript Object Notation (JSON) Pointer [4].

Table 9 – Type Error422

8.1.1.7 Type Error500

Description: Internal Server Error. (https://datatracker.ietf.org/doc/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.

Table 10 – Type Error500

8.1.1.8 Type Error501

Description: Not Implemented. (https://datatracker.ietf.org/doc/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.

Table 11 – Type Error501

8.1.2 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`

[R47] Seller **MUST** use either `X-Total-Count` or `X-Pagination-Throttled` to indicate that the page was truncated and additional results are available.

8.2 Management API Data Model

Figure 13 presents the Product Offering Qualification Management data model. The data types, requirements related to them, and mapping to MEF 79 [9] and MEF 79.0.1 [10] specifications are discussed later in this section.

The POQ Management data model is used to construct requests and responses of the API endpoints described in Section 6.1.1.

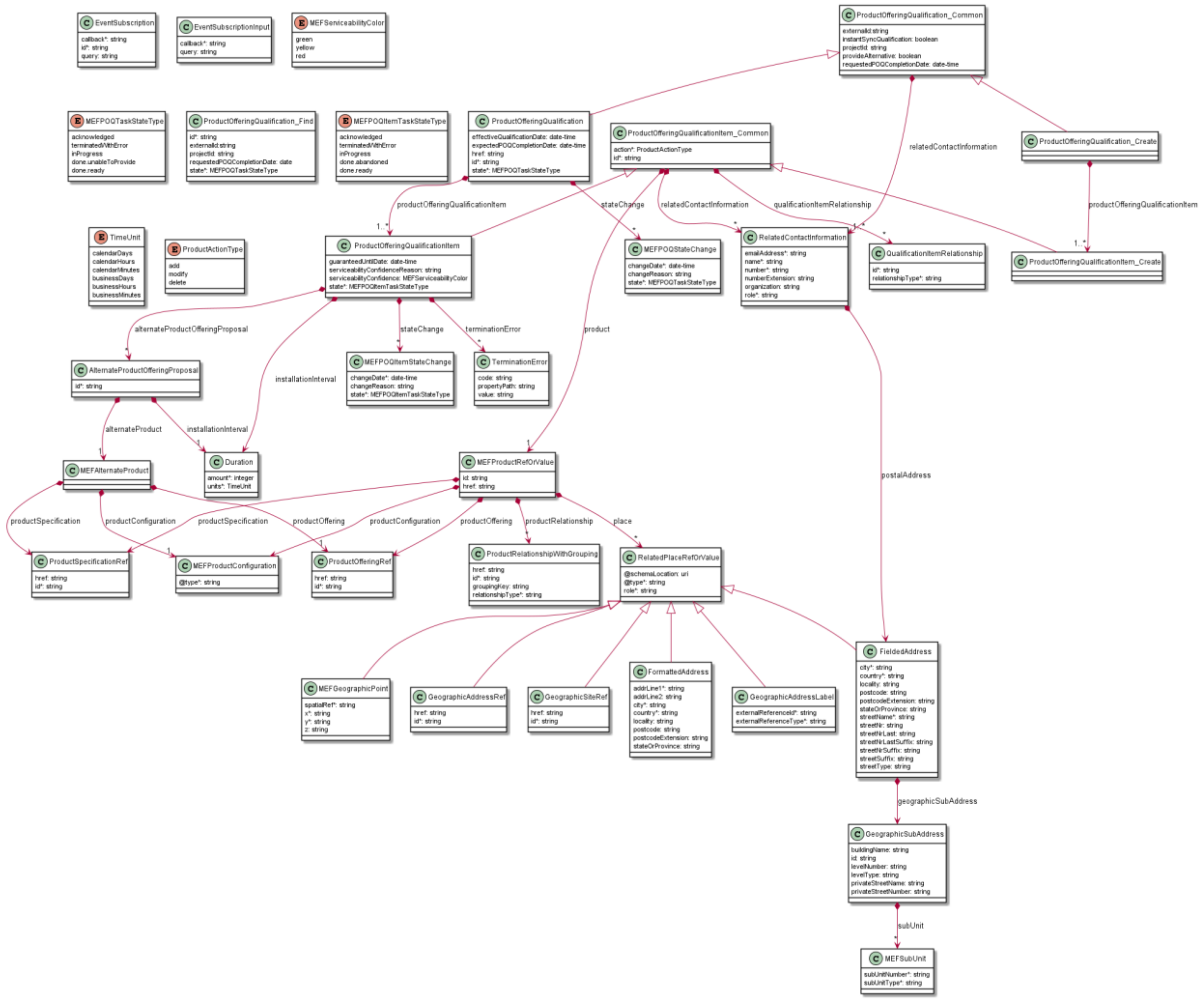

Figure 13 – Product Offering Qualification Management Data Model

8.2.1 Product Offering Qualification

8.2.1.1 Type ProductOfferingQualification_Common

Description: Defines a set of POQ attributes that might be used by the Buyer and cannot be modified by the Seller.

The `relatedContactInformation` entries provided by the Buyer cannot be changed by the Seller, however, the Seller might append related contact information to that list.

Name	Type	Description	MEF 79
`externalId`	string	ID given by the consumer and only understandable by them (to facilitate their searches afterwards).	Not represented in MEF 79
`instantSyncQualification`	boolean	If this flag is set to `true`, Buyer requests to have an instant qualification to be provided in operation POST response.	Immediate Response Only

projectId	string	This value MAY be assigned by the Buyer to identify a project the serviceability request is associated with.	Project Identifier
provideAlternative	boolean	Allows the Buyer to indicate if they are willing to get an alternate proposal if requested product not available.	Provide Alternate
relatedContactInformation*	RelatedContact Information[]	Party playing a role for this qualification. MEF 79 mandates providing 'Buyer Contact Information' in the request (role=buyerContactInformation) and 'Seller Contact Information' in the response (role=sellerContactInformation)	Allows for specifying Buyer and Seller Contact Information
requestedPOQCompletionDate	date-time	The latest date a the POQ completion is expected. This attribute is required when instantSyncQualification=false	Expected Response Date

Table 12 – Type ProductOfferingQualification_Common

8.2.1.2 Type ProductOfferingQualification_Create

Description: Represents a request formulated by the Buyer that is composed of product offering qualification items. MEF 79 Section 8.4 [9].

Inherits from:

- ProductOfferingQualification_Common

Name	Type	Description	MEF 79
productOfferingQualificationItem*	ProductOfferingQualificationItem_Create[]	A non-empty list of POQ items	Product Offering Qualification Items

Table 13 – Type ProductOfferingQualification_Create

8.2.1.3 Type ProductOfferingQualification

Description: Represents a response to the Buyer POQ inquiry. This type defines a set of attributes that are assigned by the Seller while processing the request. A POQ response is a combination of attributes defined here with common attributes that are sent in the request. This type is used in response to an immediate request and POQ retrieval by an identifier.

Inherits from:

- ProductOfferingQualification_Common

Name	Type	Description	MEF 79
effectiveQualificationDate	date-time	Date and time (set by the Seller) when the POQ state was set to one of the completion states (done.ready, done.unable_to_provide, terminated_with_error). The Seller **MUST NOT** provide this attribute until mentioned states are achieved.	Not represented in MEF 79
expectedPOQCompletionDate	date-time	The date the Seller expects to provide qualification result. Set by the Seller in case of providing a deferred response when the POQ is in acknowledged or inProgress state.	Not represented in MEF 79
href	string	Hyperlink to this POQ. Hyperlink **MAY** be used by the Seller in responses. Hyperlink **MUST** be ignored by the Seller in case it is provided by the Buyer in a request.	Not represented in MEF 79
id*	string	The Serviceability Request's unique identifier assigned by the Seller.	POQ Identifier
productOfferingQualificationItem*	ProductOffering Qualification Item[]	One or more of Product Offering Qualification Items. It **MUST** contain exactly one entry for each item in the POQ request.	Product Offering Qualification Items
state*	MEFPOQTask StateType	The state that represents the qualification status.	POQ State
stateChange	MEFPOQStateChange[]	A log of all state transitions for the POQ. It must be in sync with the most recent POQ Request state.	Not represented in MEF 79

Table 14 – Type ProductOfferingQualification

8.2.1.4 enum *MEFPOQTaskStateType*

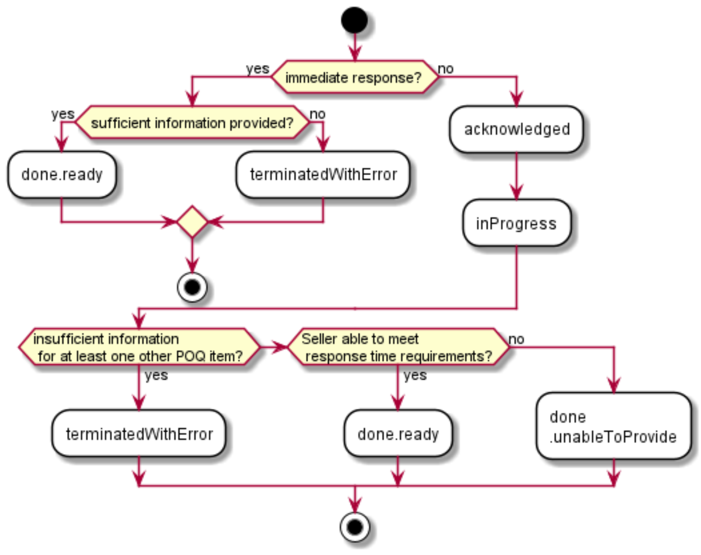

Figure 14 – POQ Activity Diagram

If a POQ request does not pass an initial validation the appropriate error response is returned to the Buyer. In case a POQ request failed business rules validation the HTTP response code is `422` and a list of validation problems is returned. Otherwise, the POQ is assigned a unique identifier. In case of a deferred response, the POQ gets the `acknowledged` state assigned and is returned in the response. In case of an immediate response the POQ moves directly to `done.ready` once the processing is done. POQ reaches `done.ready` state if all elements are in `done.ready` state as well. If an evaluation of any items concludes in state `terminatedWithError` the POQ reaches `terminatedWithError` state. The `terminatedWithError` state is reached also when there is some other information is missing in POQ request send by the Buyer. If the POQ is processed asynchronously it can reach `done.unableToProvide` if the Seller is not able to complete all items qualification by the deadline specified by the Buyer (`requestedPOQCompletionDate`).

[R48] The state of the POQ **MUST** be `done.ready` only if *all* items are in `done.ready` state. MEF 79 R46 & R68 [9].

[R49] The state of the POQ **MUST** be `terminatedWithError` only if *at least one* item is in `terminatedWithError` state. MEF 79 R47 & R69 [9].

[R50] The state of the POQ **MUST** be `inProgress` only if *at least one* item is in `inProgress` state and *none* of the items is in `terminatedWithError` state. MEF 79 R70 [9].

The following mapping has been used between `ProductOfferingQualificationStateType` and MEF 79 Section 9.1 [9]:

Description: These values represent the valid states through which the product offering qualification can transition.

Value	MEF 79	Description
acknowledged	Not represented in MEF 79	A request has been received by the Seller, has passed basic validation and the id was assigned.
terminatedWithError	INSUFFICIENT_INFORMATION_PROVIDED	This state is achieved when a well-formed POQ request has been received, but there is insufficient information to complete the PO.
inProgress	IN_PROGRESS	There is at least one POQ Item in inProgress state.
done.unableToProvide	UNABLE_TO_PROVIDE	This state is set when the Seller is unable to provide a Product Offering Qualification in the timeframe required by the Buyer.
done.ready	READY	Reached when all items are in done.ready state.

Table 15 – enum MEFPOQTaskStateType

8.2.1.5 Type MEFPOQStateChange

Description: Holds the reached state, reasons, and associated date the POQ state changed, populated by the Seller.

Name	Type	Description	MEF 79
changeDate*	date-time	The date on when the state was reached.	Not represented in MEF 79
changeReason	string	Additional comment related to state change.	Not represented in MEF 79
state*	MEFPOQTaskStateType	A state reached at change date.	Not represented in MEF 79

Table 16 – Type MEFPOQStateChange

8.2.1.6 Type ProductOfferingQualification_Find

Description: This class represent a single list item for the response of listProductOfferingQualification operation. MEF 79 Section 8.6 [9].

Name	Type	Description	MEF 79
id*	string	The POQ Request's unique identifier.	POQ Identifier
projectId	string	The project ID specified by the Buyer in the POQ Request, if any.	Project Identifier
requestedPOQCompletionDate	date	The latest date the POQ completion is expected by the Buyer, if specified by the Buyer.	Requested Response Date
state*	MEFPOQTaskStateType	Current state of the POQ.	POQ State

Table 17 – Type ProductOfferingQualification_Find

8.2.2 Product Offering Qualification Item

8.2.2.1 *Type ProductOfferingQualificationItem_Common*

Description: Common attributes shared between a POQ Item request and response. These attributes are provided by the Buyer and must not be modified by the Seller.

The `relatedContactInformation` entries provided by the Buyer cannot be changed by the Seller, however, the Seller might append related contact information to that list.

Name	Type	Description	MEF 79
`action*`	ProductActionType	Action to be qualified.	POQ Activity
`id*`	string	Id of this POQ item which is unique within the POQ. Assigned by the Buyer.	Product Offering Qualification Item Identifier
`product*`	MEFProductRefOrValue	Used by the Buyer to point to existing and/or describe the desired shape of the product. In case of `add` action – only `productConfiguration` **MUST** be specified. For `modify` action – both `id` and `productConfiguration` **MUST** be provided to point to which product instance to update and to what state. In `delete` only the `id` **MUST** be provided.	Project Identifier
`qualificationItemRelationship`	QualificationItemRelationship[]	A list of references to related POQ items in this POQ.	Requested Response Date
`relatedContactInformation`	RelatedContactInformation[]	Contact information of an individual or organization playing a role for this POQ Item (e.g. for MEF 79: POQ Item Location Contact, `role=locationContact`).	POQ State

Table 18 – Type ProductOfferingQualificationItem_Common

8.2.2.2 `enum` *ProductActionType*

Description: Action to be performed on the Product Item. The action types are described in MEF 79 Section 8.4.1.1 [9] as *POQ Activity*.

Value	MEF 79	Description
`add`	`INSTALL`	POQ item being evaluated is a new product.
`modify`	`CHANGE`	POQ item being evaluated describes a change to an existing product.
`delete`	`DISCONNECT`	POQ item is an evaluation of the feasibility of disconnecting of an existing product.

Table 19 – `enum` ProductActionType

8.2.2.3 *Type ProductOfferingRef*

Description: A reference to a Product Offering offered by the Seller to the Buyer. A Product Offering contains the commercial and technical details of a Product sold by a particular Seller. A Product Offering defines all of the commercial terms and, through association with a particular Product Specification, defines all the technical attributes and behaviors of the Product. A Product Offering may constrain the allowable set of configurable technical attributes and/or behaviors specified in the associated Product Specification. Defined in MEF 79 Section 8.4.1.1 [9].

Name	Type	Description	MEF 79
`href`	string	Hyperlink to a Product Offering in Sellers catalog. In case Seller is not providing a catalog API this field is not used. The catalog is provided by the Seller to the Buyer during onboarding. Hyperlink **MAY** be used by the Seller in responses. Hyperlink **MUST** be ignored by the Seller in case it is provided by the Buyer in a request.	Product Offering Identifier
`id*`	string	ID of a Product Offering. It is assigned by the Seller. The Buyer and the Seller exchange information about offerings' IDs during the onboarding process.	Product Offering Identifier

Table 20 – Type ProductOfferingRef

8.2.2.4 *Type QualificationItemRelationship*

Description: The relationship between product offering qualification items that can be used to validate business rules between POQ items.

Name	Type	Description	MEF 79
`id*`	string	An identifier of the targeted POQ item within the same POQ request.	Related POQ Item Identifier
`relationshipType*`	string	One of the relationship types defined in the Product Specification. For example: `UNI_REFERENCE`.	Relationship Nature

Table 21 – Type QualificationItemRelationship

8.2.2.5 *Type ProductOfferingQualificationItem_Create*

Description: This structure serves as a request for a product offering qualification item.

A product qualification item is an individual article included in a POQ that describes a Product of a particular type (Product Offering) being delivered to the geographic address or a service site specified by the Buyer.

The objective is to determine if it is feasible for the Seller to deliver this item as described and for the Seller to inform the Buyer of the estimated time interval to complete this delivery.

The modeling pattern introduces the `Common` supertype to aggregate attributes that are common to both `ProductOfferingQualificationItem` and `ProductOfferingQualificationItem_Create`. It happens that it is the Create type has a subset of attributes of the response type and does not introduce any new, thus the `Create` type has an empty definition. MEF 79 Section 8.4.1.1 [9].

Inherits from:

- ProductOfferingQualificationItem_Common

8.2.2.6 Type ProductOfferingQualificationItem

Description: An individual article included in a POQ that describes a Product of a particular type (Product Offering) being delivered to a specific geographical location. The objective is to determine if it is feasible for the Seller to deliver this item as described and for the Seller to inform the Buyer of the estimated time interval to complete this delivery. MEF 79 Section 8.4.3.1 [9].

Inherits from:

- ProductOfferingQualificationItem_Common

• Name	Type	Description	MEF 79
`alternateProductOfferingProposal`	AlternateProduct OfferingProposal[]	A list of one or more alternative Product Offerings that the Seller is proposing to the Buyer. If 1) the Buyer has set `provideAlternate=true`; 2) the Seller has determined that the POQ Confidence Level for this item is `yellow` or `red`; and 3) The Seller has alternate Products (e.g. similar but lower bandwidth) that may be adequate **MUST NOT** be specified if `state` is `terminatedWithError` or `done.abandoned`.	Alternate Product Proposals
`guaranteedUntilDate`	date-time	Date until the Seller guarantees the qualification result. **MUST NOT** be specified if `state` is `terminatedWithError` or `done.abandoned`.	Guaranteed Until Date
`installationInterval`	Duration	The estimated minimum interval that the Seller requires in their standard process to complete the delivery of this Product from the time the order is placed and any precedents have been completed. When attribute `serviceabilityConfidence` is set to `green` or `yellow` the Seller **MUST** populate this attribute. **MUST NOT** be specified if `state` is `terminatedWithError` or `done.abandoned`.	Installation Interval (Value + Unit)
`serviceabilityConfidenceReason`	string	A free text description of the reason a particular color is being provided.	Not represented in MEF 79
`serviceabilityConfidence`	MEFServiceabilityColor	The level of confidence of the Seller to be able to service the request. When the item state is `done.ready` the Seller **MUST** provide a value. It **MUST NOT** be populated for other states.	POQ Confidence Level
`state*`	MEFPOQItemTask StateType	Current state of an item.	POQ Item State

stateChange	MEFPOQItem StateChange[]	A log of all state transitions for the POQ Item. It must be in sync with the most recent POQ Item's state.	Not represented in MEF 79
terminationError	TerminationError[]	A list of text-based reasons the Seller **MUST** provide when the request cannot be processed. When item state is terminatedWithError the Seller **MUST** provide at least one termination error.	Termination Error

Table 22 – Type ProductOfferingQualificationItem

8.2.2.7 enum *MEFServiceabilityColor*

Description: A color that indicates confidence to service the request. When the item state is done.ready the Seller **MUST** provide a value. It **MUST NOT** be populated for other states.

Mapping between ServiceabilityColor and POQ Confidence Level (MEF 79 (Table 25)):

Value	MEF 79	Description
green	GREEN	The Seller has high confidence that this Product can be delivered.
yellow	YELLOW	The Seller believes they can deliver the Product but is not highly confident.
red	RED	The Seller cannot deliver the Product as specified.

Table 23 – enum MEFServiceabilityColor

8.2.2.8 enum *MEFPOQItemTaskState*

Description: Defines all possible POQ Item states.

Figure 15 depicts an activity diagram for a POQ Item lifecycle.

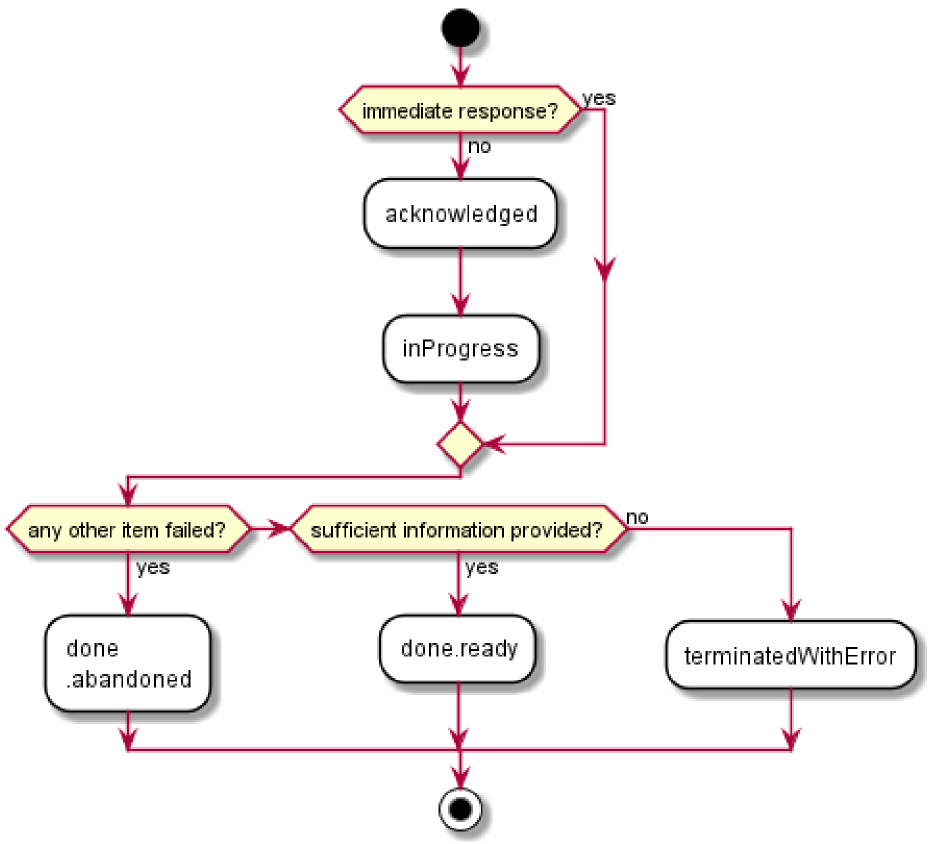

Figure 15 – POQ Item Activity Diagram

The `acknowledged` is an initial internal state of an item. The item reaches `inProgress` if the Buyer did not request instant qualification and the Seller did not provide an immediate response. If there is any other item that reached `terminatedWithError` state the currently processed item is abandoned (`done.abandoned`). If the Seller was able to successfully complete the processing of the item the `done.ready` state is assigned. Otherwise, the `terminatedWithError` state is assigned.

The following mapping has been used between `MEFPOQItemTaskStateType` and MEF 79 Section 9.2 [9]:

Value	MEF 79	Description
`acknowledged`	Not represented in MEF 79	A request has been received by the Seller and has passed basic validation.
`inProgress`	`IN_PROGRESS`	The Seller is working on a POQ item response and the answer is not ready yet.
`terminatedWithError`	`INSUFFICIENT_INFORMATION_PROVIDED`	The information provided by the Buyer is insufficient for the Seller to provide POQ Item response.
`done.abandoned`	`ABANDONED`	Applied to a POQ Item in case the final state is not reached and any other POQ Item moved to the final state other than `done`.
`done.ready`	`READY`	POQ Item response is complete. This state does not imply that Seller is able to deliver requested item.

Table 24 – enum MEFServiceabilityColor

8.2.2.9 Type MEFPOQItemStateChange

Description: Holds the reached state, reasons, and associated date the POQ state changed, populated by the Seller.

Name	Type	Description	MEF 79
changeDate*	date-time	The date on when the state was reached.	Not represented in MEF 79
changeReason	string	Additional comment related to state change.	Not represented in MEF 79
state*	MEFPOQItemTask StateType	A state which was reached at change date.	Not represented in MEF 79

Table 25 – Type MEFPOQItemStateChange

8.2.2.10 Type TerminationError

Description: This indicates an error that caused an Item to be terminated. The `code` and `propertyPath` should be used like in Error422. MEF 79 Section 8.4.3.1 [9].

Name	Type	Description	MEF 79
code	string	One of the following error codes: - `missingProperty`: The property the Seller has expected is not present in the payload - `invalidValue`: The property has an incorrect value - `invalidFormat`: The property value does not comply with the expected value format - `referenceNotFound`: The object referenced by the property cannot be identified in the Seller system - `unexpectedProperty`: Additional property, not expected by the Seller has been provided - `tooManyRecords`: the number of records to be provided in the response exceeds the Seller's threshold. - `otherIssue`: Other problem was identified (detailed information provided in a reason)	Not represented in MEF 79
propertyPath	string	A pointer to a particular property of the payload that caused the validation issue. It is highly recommended that this property should be used. Defined using JavaScript Object Notation (JSON) Pointer [4].	Not represented in MEF 79
value	string	Text to describe the reason of the termination.	Termination Error

Table 26 – Type TerminationError

8.2.3 Product Representation

8.2.3.1 Type MEFProductRefOrValue

Description: Used by the Buyer to point to existing and/or describe the desired shape of the product. In the case of the `add` action – only `productConfiguration` **MUST** be specified. For `modify` action – both id and productConfiguration **MUST** point to which product instance to update and to what state. In delete only the id **MUST** be provided. Product-related properties are described in MEF 79 Section 8.4.1.1 [9].

Name	Type	Description	MEF 79
href	string	Hyperlink to the referenced Product. Hyperlink MAY be used by the Seller in responses. Hyperlink MUST be ignored by the Seller in case it is provided by the Buyer in a request.	Not represented in MEF 79
id	string	The unique identifier of an in-service Product that is the qualification's subject. This field **MUST** be populated if an item action is either modify or delete. This field **MUST NOT** be populated if an item action is add.	Product Identifier
place	RelatedPlaceRefOrValue[]	A list of locations that are related to the Product. For example an installation location.	POQ Item Location and POQ Item Location Type
productConfiguration	MEFProductConfiguration	MEFProductConfiguration is used to specify the MEF specific product payload. This field **MUST** be populated if an item action is add or modify. It **MUST NOT** be populated when an item action is delete. The @type is used as a discriminator.	Product Specific Attributes
productOffering	ProductOfferingRef	A reference to product offering. MEF 79 defines the productOffering as mandatory for the add action, however, the API allows additionally the use case of providing the productSpecification instead. In this scenario the Buyer asks if a particular type of product can be served and gets back from the Seller direct or alternate productOfferings in response. These can be later used in Quote and Product Order steps.	Product Offering Identifier
productRelationship	ProductRelationship WithGrouping[]	A list of references to existing products that are related to the Product that would be delivered to fulfill the POQ Item.	Product Relationships
productSpecification	ProductSpecificationRef	A reference to a Product Specification used to describe the Product.	Not represented in MEF 79

Table 27 – Type MEFProductRefOrValue

Definition of place and productRelationship must comply with specific requirements of the Product Specification being used.

To indicate which Product Offering is used a Buyer may use a productOffering attribute. MEF 79 defines the productOffering as mandatory for the add action, however, the API allows additionally the use case of providing the productSpecification instead. In this scenario, the Buyer asks if a particular type of product can be served and gets back from the Seller direct or alternate productOfferings in response. These can be later used in Quote and Product Order steps.

8.2.3.2 *Type MEFProductConfiguration*

Description: MEFProductConfiguration is used as an extension point for MEF-specific product/service payload. The @type attribute is used as a discriminator.

Name	Type	Description	MEF 79
`@type*`	string	The name of the type that uniquely identifies the type of the product that is the subject of the POQ Request. In case of MEF product this is the URN provided in the Product Specification.	Not represented in MEF 79

Table 28 – Type MEFProductConfiguration

Repeated: **[R11]** `MEFProductConfiguration` type is an extension point that **MUST** be used to integrate product specifications' properties into a request/response payload.

Repeated: **[R12]** The `@type` property of `MEFProductConfiguration` **MUST** be used to specify the type of the extending entity.

Repeated: **[R35]** The product specified in `product.@type` **MUST** be compatible with the one defined by `productOffering` or `productSpecification`.

Section 8.2.2.2 describes the relation between product configuration attributes and an item action type specified by the Buyer.

8.2.3.3 Type ProductRelationshipWithGrouping

Description: A relationship to existing Product. The requirements for usage for a given Product are described in the Product Specification. The “WithGrouping” flavor of the Product Relationship allows for providing a list of related product identifiers within a single Product Relationship. This can be later used while processing the request as defined in the Product Specification. The `groupingKey` attribute is used to achieve this behavior in the API by marking the list of `ProductRelationshipWithGroupings` within a `product` with a common key.

Name	Type	Description	MEF 79
`href`	string	Hyperlink to the referenced Product. Hyperlink **MAY** be used by the Seller in responses. Hyperlink **MUST** be ignored by the Seller in case it is provided by the Buyer in a request.	Not represented in MEF 79
`id*`	string	A unique identifier of a Product that is referenced.	Related Product Identifier
`groupingKey`	string	MEF 79.0.1 Section 7 [10] introduces a list of related ids for the ProductRelationship. For sake of TMF compliance a `groupingKey` is introduced to retain `id` as a simple attribute. IDs from relationships having the same `groupingKey` and `relationshipType` **MUST** be treated as a list of identifiers.	Supports Related Product Identifier
`relationshipType*`	string	Specifies the type (nature) of the relationship to the related Product. The nature of required relationships vary for Products of different types. For example, a UNI or ENNI Product may not have any relationships, but an Access E-Line may have two mandatory relationships (related to the UNI on one end and the ENNI on the other). More complex Products such as multipoint IP or Firewall Products may have more complex relationships. As a result, the allowed and mandatory `relationshipType` values are defined in the Product Specification.	Relationship Nature

Table 29 – Type ProductRelationshipWithGrouping

MEF 79.0.1 [10] allows for providing multiple related product identifiers. This can be later used while processing the request as defined in the Product Specification. The `groupingKey` attribute is to achieve this behavior in the APIs.

[R51] The Product Identifier from relationships having the same `groupingKey` and `relationshipType` **MUST** be treated as a list of identifiers.

8.2.3.4 Type AlternateProductOfferingProposal

Description: If in the request the Buyer has requested to have alternate product proposals, then this class represents a single proposal. All properties are assigned by the Seller. MEF 79 Section 8.4.3.2 [9].

Name	Type	Description	MEF 79
`alternateProduct*`	MEFAlternateProduct	Alternate product proposal.	Related to Product Specific Attributes
`id*`	string	Identifier of the Product Offering Qualification alternate proposal. Assigned by the Seller.	Alternate Product Proposal Identifier
`installationInterval*`	Duration	The estimated minimum interval that the Seller requires in their standard process to complete the delivery of this Product from the time the order is placed and any precedents have been completed.	Installation Interval Value and Installation Interval Unit

Table 30 – Type AlternateProductOfferingProposal

8.2.3.5 Type MEFAlternateProduct

Description: An alternative Product Offering that the Seller is proposing to the Buyer. If 1) the Buyer has set `provideAlternate=true`; 2) the Seller has determined that the POQ Confidence Level for this item is `yellow` or `red`; and 3) The Seller has alternate Products (e.g. similar but lower bandwidth) that may be adequate.

Name	Type	Description	MEF 79
`productConfiguration*`	MEFProductConfiguration	Technical attributes for the Product that would be delivered to fulfill the POQ Item.	Product Specific Attributes
`productOffering*`	ProductOfferingRef	A reference to the alternate product offering.	Product Offering Identifier
`productSpecification`	ProductSpecificationRef	A reference to a Product Specification of the proposed alternate product.	Not represented in MEF 79

Table 31 – Type MEFAlternateProduct

8.2.3.6 Type ProductSpecificationRef

Description: A reference to a structured set of well-defined technical attributes and/or behaviors that are used to construct a Product Offering for sale to a market.

This data type is not represented in MEF 79.

Name	Type	Description	MEF 79
`href`	string	Hyperlink to a Product Specification in Sellers catalog. In case Seller is not providing a catalog API this field is not used. The catalog is provided by the Seller to the Buyer during onboarding. Hyperlink **MAY** be used by the Seller in responses. Hyperlink **MUST** be ignored by the Seller in case it is provided by the Buyer in a request.	Not represented in MEF 79
`id*`	string	Unique identifier of the product specification.	Not represented in MEF 79

Table 32 – Type ProductSpecificationRef

8.2.4 Place Representation

There are several formats in which place information can be introduced to the POQ request.

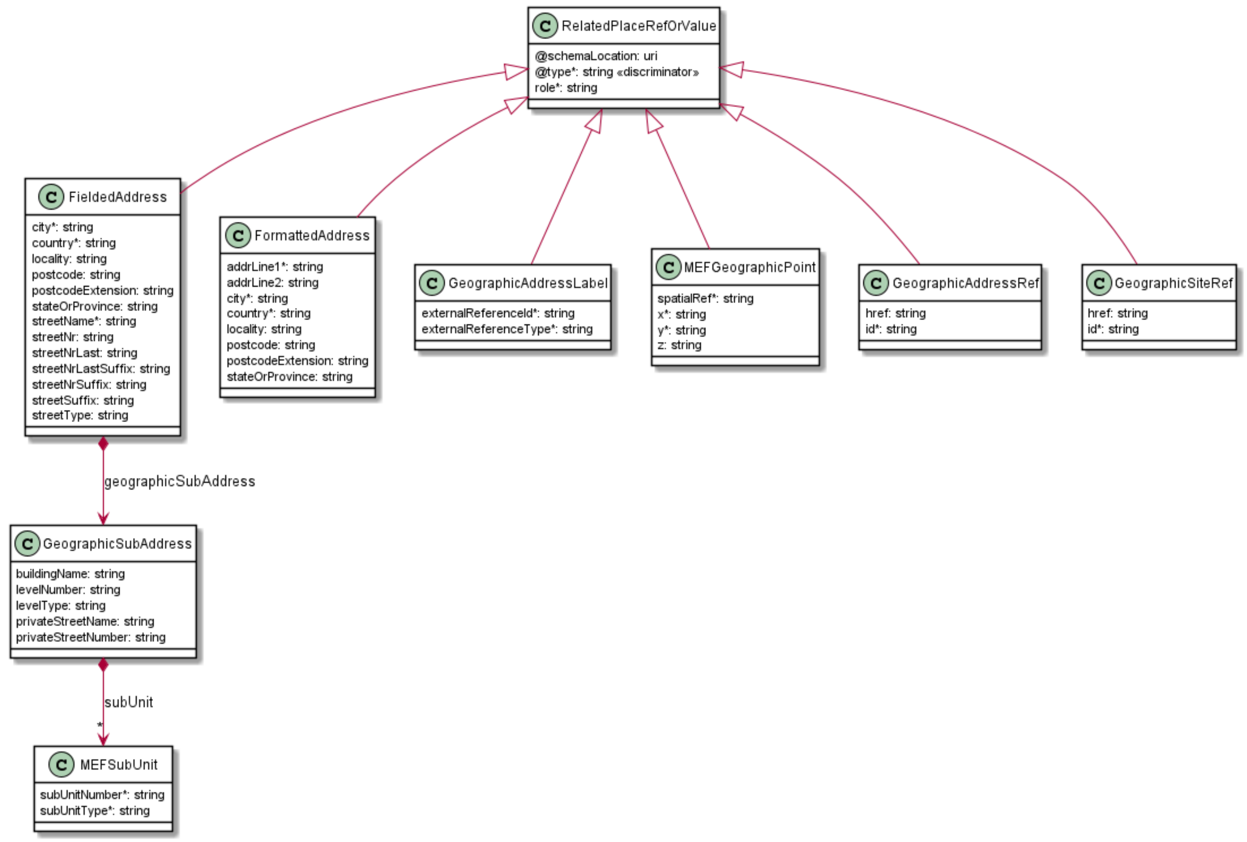

Figure 16 – Data Model Types Representing a Place

8.2.4.1 Type RelatedPlaceRefOrValue

Description: Place defines the places (locations) where the products being the subject of this qualification are to be provided.

This standard defines a number of specializations (place representations) of a place type.

Name	Type	Description	MEF 79
`@schemaLocation`	uri	A URL to a JSON-Schema file that defines additional attributes and relationships. May be used to define additional related place types. Usage of this attribute must be agreed between Buyer and Seller.	Not represented in MEF 79

@type*	string	This field is used as discriminator. The value is the name of one of the types that inherit from it using `allOf`, i.e. one of FieldedAddress, FormattedAddress, GeographicAddressLabel, MEFGeographicPoint, GeographicAddressRef, or GeographicSiteRef. Moreover, it might discriminate for additional related place as defined in `@schemaLocation`.	Relates e.g to POQ Item Location Type
role*	string	Role of this place. The values that can be specified here are described by Product Specification (e.g. `INSTALL_LOCATION`).	Not represented in MEF 79

Table 33 – Type RelatedPlaceRefOrValue

8.2.4.2 Type FieldedAddress

Description: A type of Address that has a discrete field and value for each type of boundary or identifier down to the lowest level of detail. For example "street number" is one field, "street name" is another field, etc. Reference: MEF 79 Section 8.9.2 [9].

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 79
city*	string	The city that the address is in.	City
country*	string	The country that the address is in.	Country
geographicSubAddress	GeographicSubAddress	Additional fields used to specify an address, as detailed as possible.	Not represented in MEF 79
locality	string	The locality that the address is in.	Locality
postcode	string	A descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as zip code). MEF 79 defines it as required however as in certain countries it is not used we make it optional in the API.	Postal Code
postcodeExtension	string	An extension of a postal code. E.g. the part following the dash in a US urban property address.	Postal Code Extension
stateOrProvince	string	The State or Province that the address is in.	State Or Province
streetName*	string	Name of the street or other street type.	Street Name
streetNr	string	Number identifying a specific property on a public street. It may be combined with `streetNrLast` for ranged addresses. MEF 79 defines it as required for the Seller response, however in certain countries it is not used, so it's optional in the API.	Street Number
streetNrLast	string	Last number in a range of street numbers allocated to a property.	Street Number Last
streetNrLastSuffix	string	Last street number suffix for a ranged address.	Street Number Suffix Last
streetNrSuffix	string	The first street number suffix.	Street Number Suffix
streetSuffix	string	A modifier denoting a relative direction.	Street Suffix
streetType	string	The type of street (e.g., alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf).	Street Type

Table 34 – Type FieldedAddress

8.2.4.3 Type FormattedAddress

Description: A type of Address that has discrete fields for each type of boundary or identifier with the exception of the street and more specific location details, which are combined into a maximum of two strings based on local postal addressing conventions. Reference: MEF 79 Section 8.9.3 [9].

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 79
addrLine1*	string	The first address line in a formatted address.	Address Line 1
addrLine2	string	The second address line in a formatted address.	Address Line 2
city*	string	The city that the address is in.	City
country*	string	The country that the address is in.	Country
locality	string	An area of defined or undefined boundaries within a local authority or other legislatively defined area, usually rural or semi-rural in nature.	Locality
postcode	string	A descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as ZIP code).	Postal Code
postcodeExtension	string	An extension of a postal code. E.g. the part following the dash in a US urban property address.	Postal Code Extension
stateOrProvince	string	The State or Province that the address is in.	State Or Province

Table 35 – Type FormattedAddress

8.2.4.4 Type MEFGeographicPoint

Description: A MEFGeographicPoint defines a geographic point through coordinates. Reference: MEF 79 Section 8.9.5 [9].

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 79
spatialRef*	string	The spatial reference system used to determine the coordinates (e.g. “WGS84”). The system used and the value of this field are to be agreed during the onboarding process.	Spatial Reference
x*	string	The latitude expressed in the format specified by the spacialRef.	Latitude
y*	string	The longitude expressed in the format specified by the spacialRef.	Longitude
z	string	The elevation expressed in the format specified by the spacialRef.	Elevation

Table 36 – Type MEFGeographicPoint

[R52] The spatialRef value to be used **MUST** be agreed between Buyer and Seller during the onboarding process.

8.2.4.5 Type GeographicSubAddress

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

Name	Type	Description	MEF 79
buildingName	string	Allows for identification of places that require building name as part of addressing information.	Building Name
levelNumber	string	Used where a level type may be repeated e.g. BASEMENT 1, BASEMENT 2.	Level Number
levelType	string	Describes level types within a building.	Level Type
privateStreetName	string	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	Private streets numbers internal to a private street.	Private Street Number
subUnit	MEFSubUnit[]	Representation of a MEFSubUnit. It is used for describing subunit within a subAddress e.g. BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.	Sub Unit List

Table 37 – Type GeographicSubAddress

8.2.4.6 Type GeographicAddressRef

Description: A reference to a Geographic Address resource available through Sonata Address Validation API. Identifier from MEF 79 (Table 21) [9].

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 79
href	string	Hyperlink to the referenced Address. Hyperlink **MAY** be used by the Seller in responses. Hyperlink **MUST** be ignored by the Seller in case it is provided by the Buyer in a request.	Not represented in MEF 79
id*	string	Identifier of the referenced Geographic Address. This identifier is assigned during a successful address validation request (Sonata Geographic Address Management API).	Fielded | Formatted | Geographic Address Label | Geographic Point Identifier

Table 38 – GeographicAddressRef

8.2.4.7 Type GeographicSiteRef

Description: A reference to a Geographic Site resource available through Sonata Service Site API. Identifier from MEF 79 (Table 21) [9].

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 79
href	string	Hyperlink to the referenced Site. Hyperlink **MAY** be used by the Seller in responses. Hyperlink **MUST** be ignored by the Seller in case it is provided by the Buyer in a request.	Not represented in MEF 79
id*	string	Identifier of the referenced Geographic Site.	Site Identifier

Table 39 – GeographicSiteRef

8.2.4.8 Type GeographicAddressLabel

Description: A unique identifier controlled by a generally accepted independent administrative authority that specifies a fixed geographical location. Reference: MEF 79 Section 8.9.4 [9].

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 79
externalReferenceId*	string	The unique reference to an Address as provided by the Administrative Authority.	Administrative Authority Address Label
externalReferenceType*	string	The organization or standard from the organization that administers this Geographic Address Label ensuring it is unique within the Administrative Authority. The value(s) to be used are to be agreed during the onboarding. For North American providers this would normally be CLLI (Common Language Location Identifier) code.	Administrative Authority

Table 40 – GeographicAddressLabel

8.2.4.9 Type MEFSubUnit

Description: A Sub Unit type. Reference: MEF 79 Section 8.9.2 [9].

Name	Type	Description	MEF 79
subUnitNumber*	string	The discriminator used for the subunit, often just a simple number but may also be a range.	Sub Unit Name
subUnitType*	string	The type of subunit e.g. BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.	Sub Unit Type

Table 41 – Type MEFSubUnit

8.2.5 Notification Registration

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

8.2.5.1 Type EventSubscriptionInput

Description: This class is used to register for Notifications.

Name	Type	Description	MEF 79
`callback*`	string	This callback value must be set to *host* property from the Buyer ProductOfferingQualification Notification API (`productOfferingQualificationNotification.api.yaml`).This property is appended with base path and notification resource path specified in that API to construct an URL to which the notification is sent. E.g. for "callback": `https://buyer.co/listenerEndpoint`, the create event notification will be sent to: `https://buyer.co/listenerEndpoint/mefApi/sonata/productOfferingQualificationNotification/v7/listener/poqCreateEvent`	Return Address Information
`query`	string	This attribute is used to define to which type of events to register to. Example: `query`:`eventType = poqCreateEvent`. To subscribe for more than one event type, put the values separated by comma: `eventType=poqCreateEvent,poqStateChangeEvent`. The possible values are enumerated by the `POQEventType` in `productOfferingQualificationNotification.api.yaml`. An empty query is treated as specifying no filters – ending in subscription for all event types.	List of Notification Types

Table 42 – Type EventSubscriptionInput

The `query` formatting complies to RCF 3986 [3]. According to it, every attribute defined in the Event model (from the notification API) can be used in the `query`. However, this standard requires only the `eventType` attribute to be supported.

[R53] `eventType` is the only attribute that the Seller **MUST** support in the query.

8.2.5.2 Type EventSubscription

Description: This resource is used to manage notification subscriptions. Reference: MEF 79 Section 8.3 [9].

Name	Type	Description	MEF 79
`callback*`	string	The value provided by the Buyer in `EventSubscriptionInput` during notification registration.	Return Address Information
`id*`	string	An identifier of this Event Subscription assigned by the Seller when resource is created.	Not represented in MEF 79
`query`	string	The value provided by the Buyer in `EventSubscriptionInput` during notification registration.	List of Notification Types

Table 43 – Type EventSubscription

8.2.6 Common

8.2.6.1 Type RelatedContactInformation

Description: Contact data for a person or organization that is involved in the product offering qualification. In a given context it is always specified by the Seller (e.g. Seller Contact Information) or by the Buyer. Reference: MEF 79 Section 8.11 [9].

Name	Type	Description	MEF 79
emailAddress*	string	Email address	Contact Email Address
name*	string	Name of the contact	Contact Name
number*	string	Phone number	Contact Phone Number
numberExtension	string	Phone number extension	Contact Phone Number Extension
organization	string	The organization or company that the contact belongs to	Contact Organization
role*	string	A role of the particular contact in the request	Not represented in MEF 79
postalAddress	FieldedAddress	Identifies the postal address of the person or office to be contacted.	Not represented in MEF 79

Table 44 – Type RelatedContactInformation

The related contact information can be defined at a POQ or a POQ Item level. In both cases, it is allowed to provide a list of party role information. The `role` attribute is used to provide a reason the particular party information is used. It can result from MEF 79 requirements (e.g. Seller Contact Information) or from the Product Specification requirements.

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

- Seller Contact Information: `role=sellerContactInformation`
- Buyer Contact Information: `role=buyerContactInformation`

8.2.6.2 Type Duration

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

Name	Type	Description	MEF 79
amount*	integer	Duration (number of seconds, minutes, hours, etc.)	Duration Value
units*	TimeUnit	Time unit type	Duration Unit

Table 45 – Type Duration

8.2.6.3 enum *TimeUnit*

Description: Represents a unit of time. Reference: MEF 79 Sections 8.4.3.1 and 8.4.3.2 [9].

Value	MEF 79
calendarMonths	CALENDAR_MONTHS
calendarDays	CALENDAR_DAYS
calendarHours	CALENDAR_HOURS
calendarMinutes	CALENDAR_MINUTES
businessDays	BUSINESS_DAYS
businessHours	BUSINESS_HOURS
businessMinutes	BUSINESS_MINUTES

Table 46 – Type TimeUnit

[R54] The clarification of what Business days, hours, and minutes mean **MUST** be done between the Buyer and the Seller during the onboarding process.

8.3 Notification API Data Model

Figure 17 presents the Product Offering Qualification Notification data model. The data types, requirements related to them and mapping to MEF 79 and MEF 79.0.1 specifications are discussed later in this section.

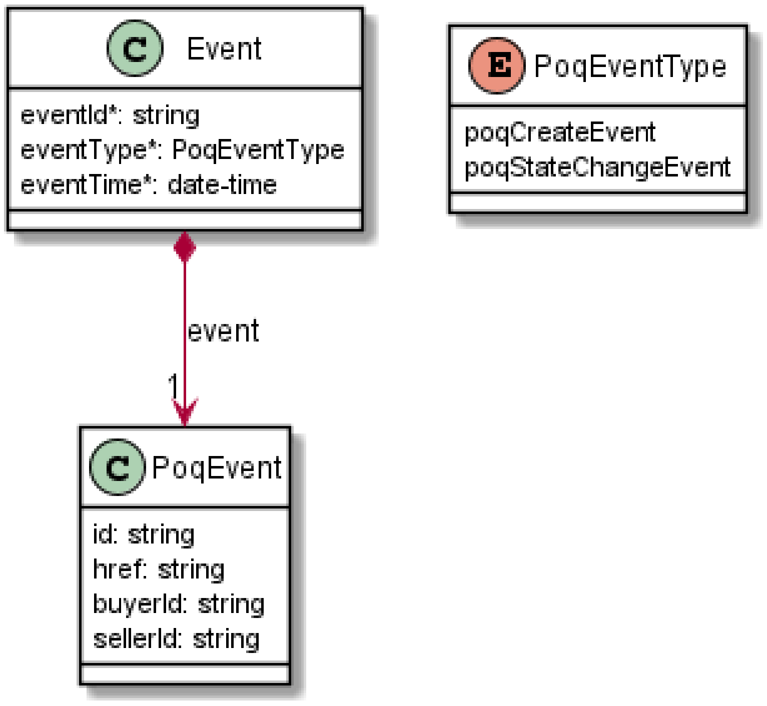

Figure 17 – Product Offering Qualification Notifications Data Model

The POQ Management data model is used to construct requests and responses of the API endpoints described in Section 6.1.2.

8.3.1 Type Event

Description: Event class is used to describe information structure used for notification. Reference: MEF 79 Section 8.5 [9].

Name	Type	Description	MEF 79
eventId*	string	Id of the event.	Not represented in MEF 79
eventType*	POQEventType	Indicates the type of the POQ event.	State Change Type
eventTime*	date-time	Datetime when the event occurred.	Not represented in MEF 79
event*	POQEvent	A reference to the POQ that is source of the notification.	Not represented in MEF 79

Table 47 – Type Event

8.3.2 Type POQEvent

Description: A reference to the POQ that is the source of the notification. Reference: MEF 79 Section 8.8 [9]

Name	Type	Description	MEF 79
id*	string	The POQ Request's unique identifier.	POQ Identifier
href	string	Link to the POQ.	Not represented in MEF 79
buyerId	string	The unique identifier of the organization that is acting as the Buyer. **MUST** be specified in the request only when the responding represents more than one Buyer.	Buyer ID
sellerId	string	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.	Seller ID

Table 48 – Type POQEvent

8.3.3 enum POQEventType

Description: Indicates the type of product offering qualification event.

Value	MEF 79
poqCreateEvent	CREATE
poqStateChangeEvent	CHANGE
poqItemStateChangeEvent	CHANGE

Table 49 – Type POQEventType

9 References

[1] IETF JSON Schema draft 7, *JSON Schema: A Media Type for Describing JSON Documents* and associated documents, by Austin Wright and Henry Andrews, March 2018. Copyright © 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

[2] IETF RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*, March 1997

[3] IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, by Tim Berners-Lee and Roy T. Fielding and Larry M Masinter, January 2005. Copyright © The Internet Society (2005).

[4] IETF RFC 6901, *JavaScript Object Notation (JSON) Pointer*, by Paul C. Bryan and Kris Zyp and Mark Nottingham, April 2013. Copyright © 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

[5] IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, by Roy T. Fielding and Julian Reschke, June 2014. Copyright © 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

[6] IETF RFC 8174, *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*, May 2017

[7] Fielding, Roy Thomas, *Architectural Styles and the Design of Network-based Software Architectures (Ph.D)*, 2000

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

[9] MEF 79, *Address, Service Site, and Product Ordering Qualification Management – Requirements and Use Cases*, November 2019

[10] MEF 79.0.1, *Amendment to MEF 79: Address, Service Site, and Product Offering Qualification Management Requirements and Use Cases*, December 2020

[11] MEF 79.0.2, *Amendment to MEF 79: Address Validation*, July 2021

[12] MEF 80, *Quote Management Requirements and Use Cases*, July 2021

[13] OpenAPI Initiative, *OpenAPI Specification (OAS) v3.0.3*, February 2020

[14] TMF679 TM Forum, *TMF679 Product Offering Qualification API Rest Specification R19.0.1*, June 2019