
# Working Draft

# LSO Payload Handbook

## June 2022

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

Disclaimer



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

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

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

a) any express or implied license or right to or under any patent, copyright, trademark or trade secret rights held or claimed by any MEF member which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor

b) any warranty or representation that any MEF members will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor

c) any form of relationship between any MEF member and the recipient or user of this document.

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

# Table of Contents

## List of Figures



## List of Tables

# 1 Abstract

The aim of this document is to provide a comprehensive guide for developers of both MEF and Non-MEF LSO Payloads (product and service schemas) by describing the governance, technical and documentation requirements. It also describes the procedure of publishing the Non-MEF Payload schemas in the MEF's LSO Marketplace.

## 2 Terminology and Abbreviations

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

| Term | Definition | Reference |
|---|---|---|
| **IRP** | **Interface Reference Point**<br>The logical point of interaction between specific management entities | MEF 55.1 [4] |
| **LSO** | **Lifecycle Service Orchestration**<br>Open and interoperable automation of management operations over the entire lifecycle of Services. This includes fulfillment, control, performance, assurance, usage, security, analytics, and policy capabilities, for the network domains that require coordinated management and control to deliver the Service. | MEF 55.1 [4] |
| **LSO Envelope** | **LSO Envelope API**<br>API (Function-specific information and Function-specific operations) capable of carrying Product or Service specific information. (e.g. Product Order API) | This document |
| **LSO Marketplace** | MEF microsite optimized for accessibility of all LSO API-related information and resources for both technical and business audiences. | lso.mef.net |
| **LSO Payload** | Product or Service specific information that is exchanged with LSO Envelope API | This document |
| **MEF-Endorsed LSO Payload** | A non-MEF LSO Payload that has been documented according to the requirements in this document and has been approved by the MEF membership for inclusion in the LSO Marketplace | This document |
| **MEF-Standardized LSO Payload** | A schema for use in LSO APIs that is included in a MEF Standard and based on other MEF standards. | This document |
| **Non-MEF LSO Payload** | A schema for use in LSO APIs that is included in the LSO Marketplace but which is not based on MEF standards. | This document |
| **Partner-Specific LSO Payload** | A non-MEF LSO payload that has not been documented according to the requirements in this document, but nonetheless has been approved by the MEF membership for inclusion in the LSO Marketplace. Partner in this context refers to the counter-party of the LSO API implementer (e.g. telecom supplier, telecom customer, enterprise customer, etc.) | This document |

| Term | Definition | Reference |
|---|---|---|
| **POQ** | **Product Offering Qualification** | MEF 79 [6] |
| **SDO** | **Standards Developing Organization** | |
| **URI** | **Uniform Resource Identifier**<br>A compact string of characters for identifying an abstract or physical resource classified as a locator, a name, or both | RFC 3986 [12] |
| **URN** | **Uniform Resource Names**<br>A subset of URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable. | RFC 8141 [11]<br><br>RFC 3986 [12] |

# 3 Introduction

MEF has defined standardized APIs to automate business and operational interactions between Buyers and Sellers based on the LSO Reference Architecture (MEF 55.1 [4]). The LSO Reference Architecture identifies the management Interface Reference Points (LSO Interface Reference Points (IRP)), the logical points of interaction between specific functional management components. The LSO Reference Architecture is presented in Figure 1:

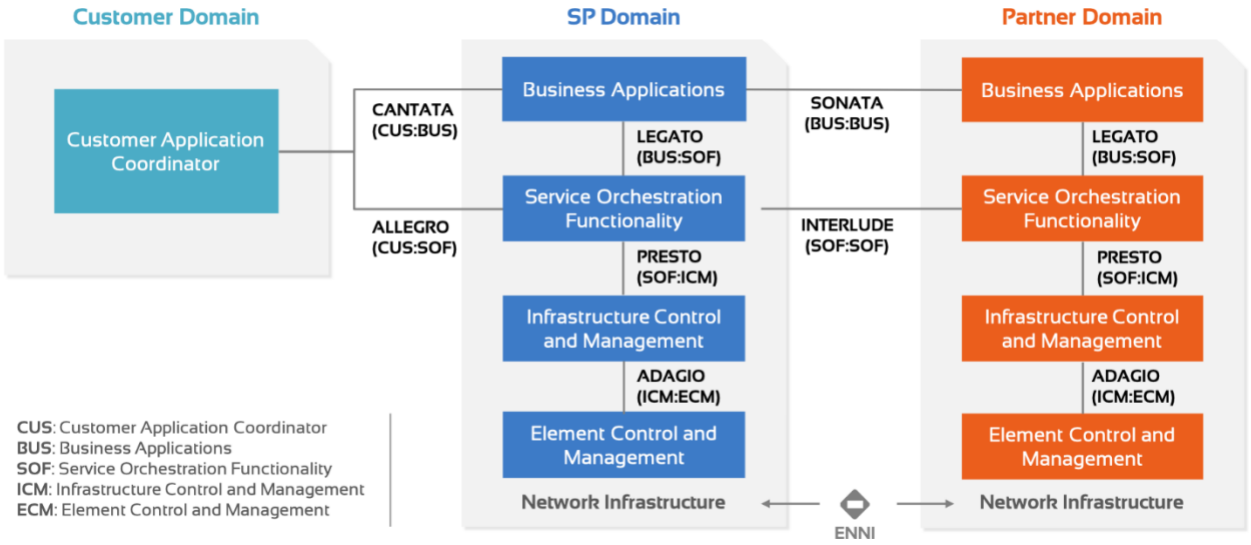



**Figure 1 – LSO Reference Architecture**

Each of the IRPs is implemented by a set of LSO APIs associated with the IRP's name. LSO APIs comprise two structural components:

- Product or Service independent information (Function-specific information and Function-specific operations). This part is referred to as the LSO API Envelope ("**Envelope**").
- Product or Service-specific information (carrying MEF product or service specification data model). This part is referred to as the product or service LSO Payload ("**Payload**").

## 3.1 MEF-Standardized LSO Payloads

Three of the IRPs interfacing with Service Orchestration Functionality (LSO Legato, LSO Allegro, LSO Interlude) support operational actions to order and configure a **service** in the network. Two of the IRPs interfacing with Business Applications (LSO Cantata and LSO Sonata) support the business actions on a **product** where a product is a business abstraction of a service. MEF standardizes both product and service specifications. These detailed standardized specifications are also represented as schemas that can be used together with Envelopes. These schemas are referred to as 'MEF-Standardized LSO Payloads', and form the first category of LSO Payloads.

## 3.2 Non-MEF LSO Payloads

By virtue of the product/service independent nature of the LSO APIs, it is possible to develop schemas not based on MEF standards and successfully use those schemas in LSO APIs as well. Such schemas are referred to as 'Non-MEF LSO Payloads' and form the second category of LSO Payload.

Non-MEF LSO Payloads may be important to implementers of LSO APIs for a variety of commercial reasons. These implementers may want to use the LSO Marketplace to broaden the awareness of their Non-MEF LSO Payloads.

It may be that a product or service is not a MEF-Standardized LSO Payload for one or more of the following reasons:

- The expertise required for specifying the product or service in question does not exist in MEF (e.g. voice, power, mobility, etc.)
- It may already have been addressed in another SDO.
- There is not enough market justification for MEF members to commit resources to the development of the standardization of the product or service in question. However, it may be that traction achieved with the MEF-Endorsed Payload will trigger the demand and resources need to start a regular MEF standardization project at a later date.
- The standardization of the product or service in MEF will take longer than the window of opportunity, and by the time the standardized payload is available, the market opportunity will have passed.

This document is designed to be useful for developers of both MEF and non-MEF LSO Payloads that want to have their schemas included in the MEF's LSO Marketplace.

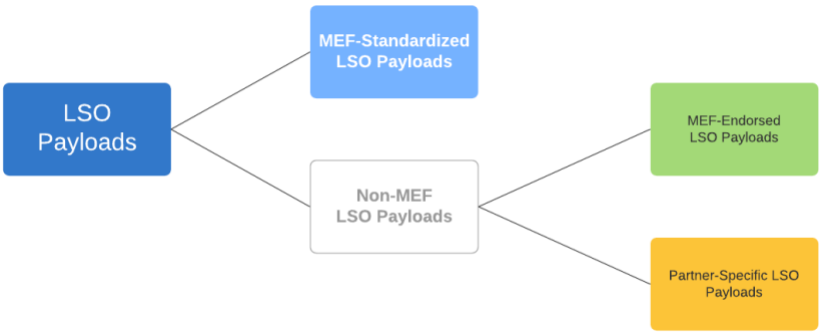



**Figure 2 – Categories of LSO Payloads**

There are two sub-categories of non-MEF LSO Payloads:

### 3.2.1 MEF-Endorsed LSO Payloads

These are schemas that meet the following requirements:

- Meet technical requirements described in Section 5
- Meet the documentation requirements described in Section 6
- Specifically approved for inclusion in the LSO Marketplace by the MEF membership as a 'MEF-Endorsed LSO Payload' as described in Section 4.

Typically, but not exclusively, MEF-Endorsed LSO Payloads are derived from standards developed, or being developed, in other SDOs.

### 3.2.2 Partner-Specific LSO Payloads

These are schemas that meet the following requirements:

- Meet technical requirements described in Section 5
- Specifically approved for inclusion in the LSO Marketplace by the MEF membership as 'Partner-Specific LSO Payload' as described in Section 4.

## 3.3 Choosing an LSO Payload Goal

Below are the general considerations and steps:

- Check the MEF Standardized payload roadmaps in the LSO Marketplace. Your product or service may be a subject of an open project. In that case, you should verify in detail whether it suits your needs. You can join the project and contribute by expressing your requirements if needed.
- Complete the Proposal Form (see Appendix B Proposal Form for Non-MEF LSO Product Payload). If the MEF Standard path cannot be applied, this step is needed to get the initial MEF review and acceptance of your proposal. This is to avoid a situation in which the new payload would overlap with any other existing or developed LSO Payload or is not breaking any MEF best practices.
- Read this Handbook. It provides requirements and best practices on how to build and ship a payload schema.
- Assess the desired Payload type (MEF-Standardized, MEF-Endorsed, Partner-Specific)
- Follow the steps as described in section 4.1 or 4.2 and build your schema(s).

After passing the Assessment and Approval – your payload schemas will be published on the LSO Marketplace.

# 4 Governance

This section describes the process and rules for establishing Non-MEF LSO Payloads in the LSO Marketplace. For reference purposes, additional information on the establishment of MEF-Standardized LSO Payloads in the LSO Marketplace is provided in Appendix C.

## 4.1 MEF-Endorsed LSO Payloads

The process for a MEF member (or a non-MEF member partnered with a MEF member for this purpose) to establish a MEF-Endorsed LSO Payload in the LSO Marketplace is as follows:

- MEF member reviews this document to ensure that it can meet all the technical (Section 0) and documentation requirements (Section 6)
- MEF member submits a proposal together with a Payload schema (see Appendix B)
- MEF CBC electronic ballot is held for two weeks to see if there is a majority in favor of continuing the proposal process. A simple majority is needed.
- If the decision is to continue, the MEF will facilitate the testing of the proposed LSO Payload for successful binding in accordance with the technical requirements in Section 0.
- If the testing is successful, the MEF member completes the documentation according to Section 6.
- Once the documentation is complete, the proposal together with the documentation, the schema, and results of the testing are submitted to the MEF LSO Committee
- MEF LSO Committee holds a two-week electronic ballot to confirm that the submission is sufficiently high in quality to merit the moniker 'MEF-Endorsed LSO Payload' and inclusion in the LSO Marketplace.
- If the proposal passes the ballot, the LSO Payload is included in the LSO Marketplace as a MEF-Endorsed LSO Payload.
- Note that MEF-Endorsed LSO Payloads remain available in the LSO Marketplace as long as all the following criteria are met:
    - The submitter of the MEF-Endorsed LSO Payload has not requested its removal
    - The submitter of the MEF-Endorsed LSO Payload is a current MEF member
    - No proposal by a MEF member to remove the MEF-Endorsed LSO Payload from the LSO Marketplace has been made to the LSO Committee and passed in a procedural motion in the LSO Committee


**Figure 3 – MEF-Endorsed LSO Payload development processes**

## 4.2 Partner-Specific LSO Payloads

The process for a MEF member (or a non-MEF member partnered with a MEF member for this purpose) to create such a Partner-Specific LSO Payload is as follows:

- MEF member reviews this document to ensure that it can meet all the technical (Section 0) and documentation requirements (Section 6)

- MEF member submits a proposal together with a Payload schema (see Appendix B)
- MEF CBC electronic ballot is held for two weeks to see if there is a majority in favor of continuing the proposal process. A simple majority is needed.
- If the decision is to continue, the MEF will facilitate the testing of the proposed LSO Payload for successful binding in accordance with the technical requirements in Section 5.
- If the testing is successful, the proposal together with the schema and results of the testing are submitted to the MEF LSO Committee for review during a two-week electronic ballot, the purpose of which is to confirm that the payload schema is sufficiently high in quality to merit the moniker 'Partner-Specific LSO Payload' and inclusion in the LSO Marketplace.
- If the proposal passes the ballot, the LSO Payload is included in the LSO Marketplace as a Partner-Specific LSO Payload.
- Note that Partner-Specific LSO Payloads remain available in the LSO Marketplace as long as all the following criteria are met:
    - The submitter of the MEF-Endorsed LSO Payload has not requested its removal
    - The submitter of the Partner-Specific LSO Payload is a current MEF member
    - No proposal by a MEF member to remove the Partner-Specific LSO Payload from the LSO Marketplace has been made to the LSO Committee and passed in a procedural motion in the LSO Committee

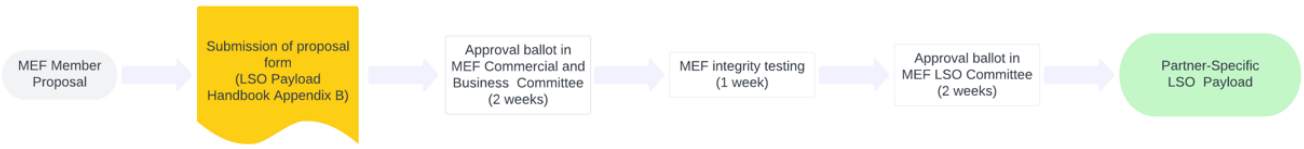



**Figure 4 – Partner-Specific Payload development processes**

## 4.3 Comparison of LSO Payload Development Processes

Figure 5 shows all three development processes for LSO Payloads together to illustrate the differences in approaches for the three categories.


**Figure 5 – LSO Payload development processes**

# 5 Non-MEF LSO Payload Technical Requirements

This section describes the technical requirements that need to be met for a Payload to be used successfully with an Envelope. It covers the following topics:

- Integration of the Payload into the Envelope
- Payload format
- Naming conventions
- API flavors
- "$id"/URN structure
- Common Model reuse
- Internal product/service dependencies
- Payload-related Envelope requirements
- Packaging
- Static and dynamic binding
- Binding tool

The technical requirements are described with the use of two examples:

- Simple Product
- Complex Product

*Note:* Illustrations are given for a product specification rather than for service specification. The same requirements and rules apply for service specifications (unless explicitly stated otherwise). Also for clarity of the text, the term "product" is used to mean "product or service" (unless explicitly stated otherwise).

Figure 6 and Figure 7 present Simple Product and Complex Product payload models.

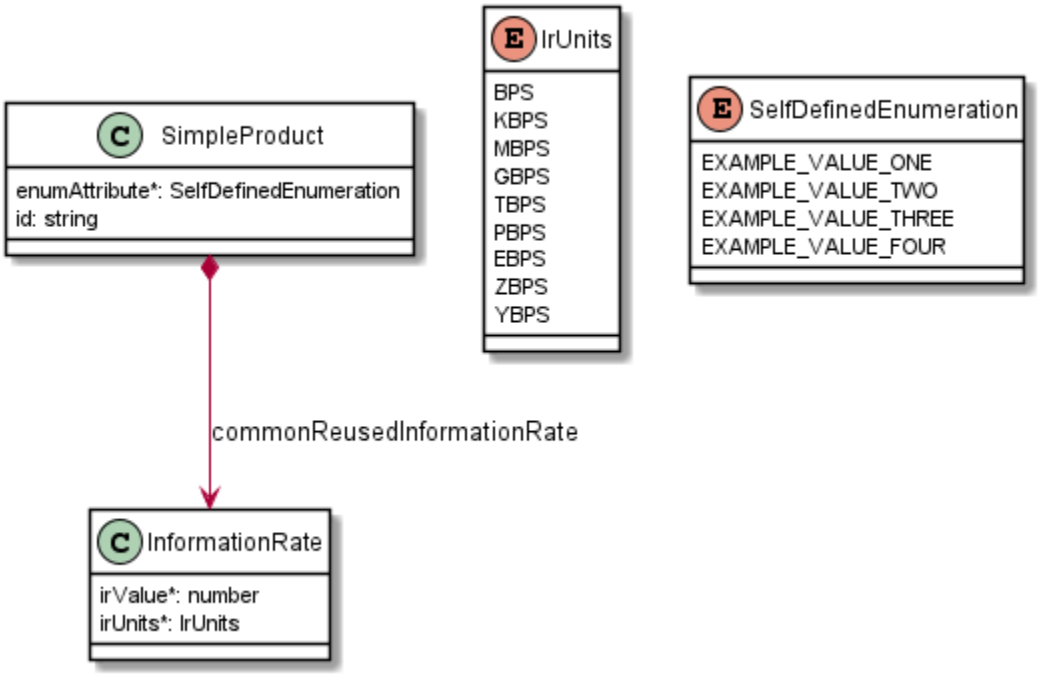



**Figure 6 – Simple Product model**

Simple Product presents the basic concepts of:

- simple string attribute (`id`)
- definition of an enumeration (`enumAttribute`, `SelfDefinedEnumeration`)
- reference attribute using MEF common type (`commonReusedInformationRate`. `InformationRate`)

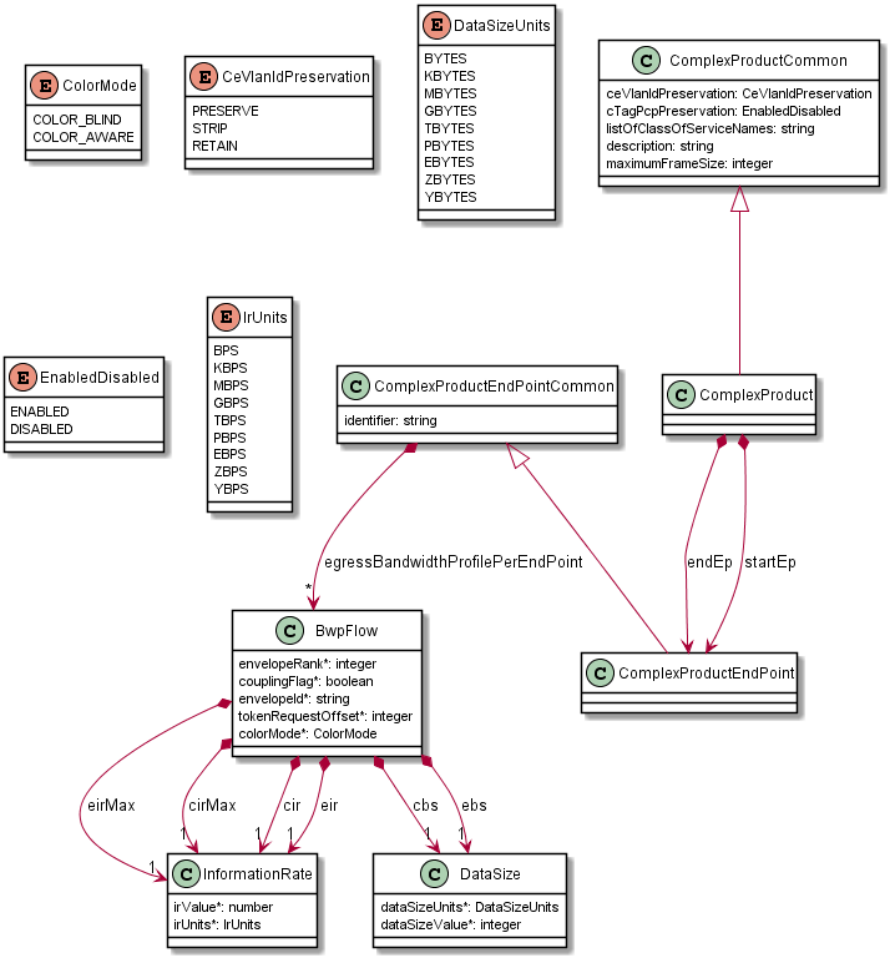



**Figure 7 – Complex Product model**

Complex Product additionally presents:

- Providing various flavors per API function
- Extracting Common class that allows easy definition of required attribute list per function.
- Reusing more complex MEF common types

## 5.1 Binding of LSO Payload with the LSO Envelope APIs

LSO APIs are product/service-agnostic in the sense that they provide business interactions between the Buyer and the Seller and they do not contain any product-specific information in their specifications. To pass the product-specific information, an extension pattern must be used. This applies only to APIs that carry product-specific information. In Sonata IPR these are Product Offering Qualification, Quote, Product Order, and Product Inventory. Address Validation, Site, and Trouble Ticket do not carry product-specific information.

The extension hosting type in the envelope API data model is `MEFProductConfiguration`. The `@type` attribute of that type must be set to a value that uniquely identifies the product specification (Figure 8). This identifier is specified in the `$id` field of the root payload schema (section 8.2 of [1]). In this document's example schemas this will be (the details of the "$id" and the urn format are explained in section 5.5):

- `"$id": urn:mef:lso:spec:sonata:simple-product:v1.0.0:all`
- `"$id": urn:mef:lso:spec:cantata-sonata:complex-product:v1.0.0:poq`

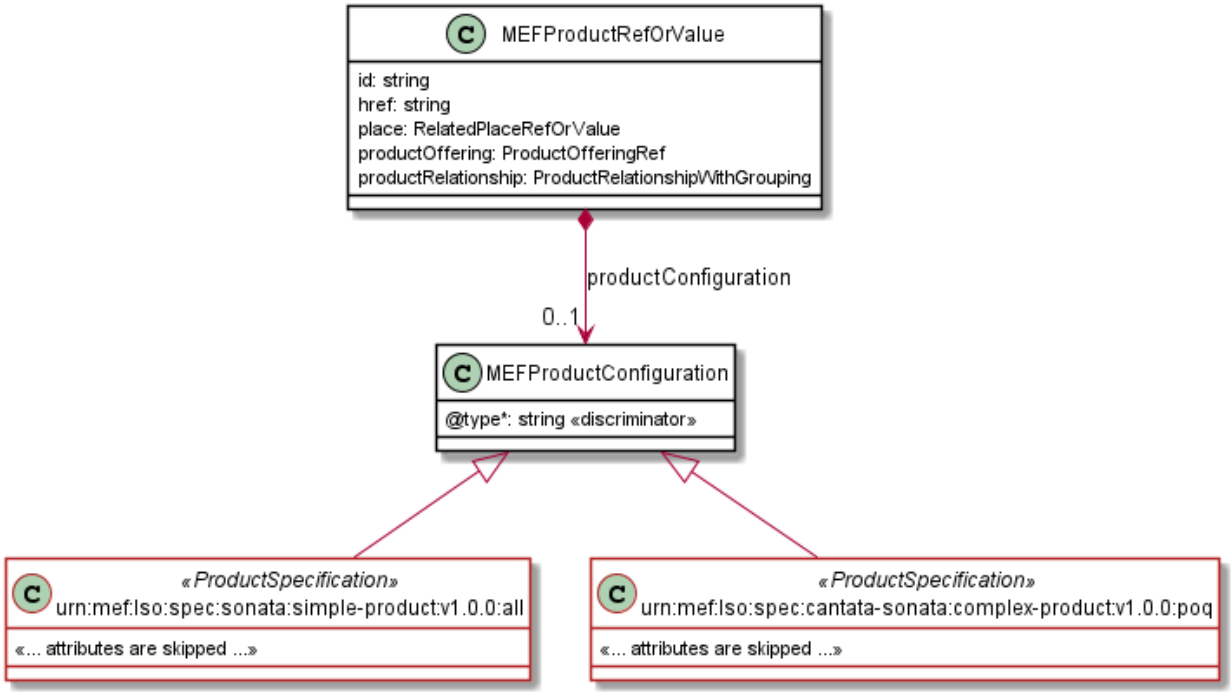
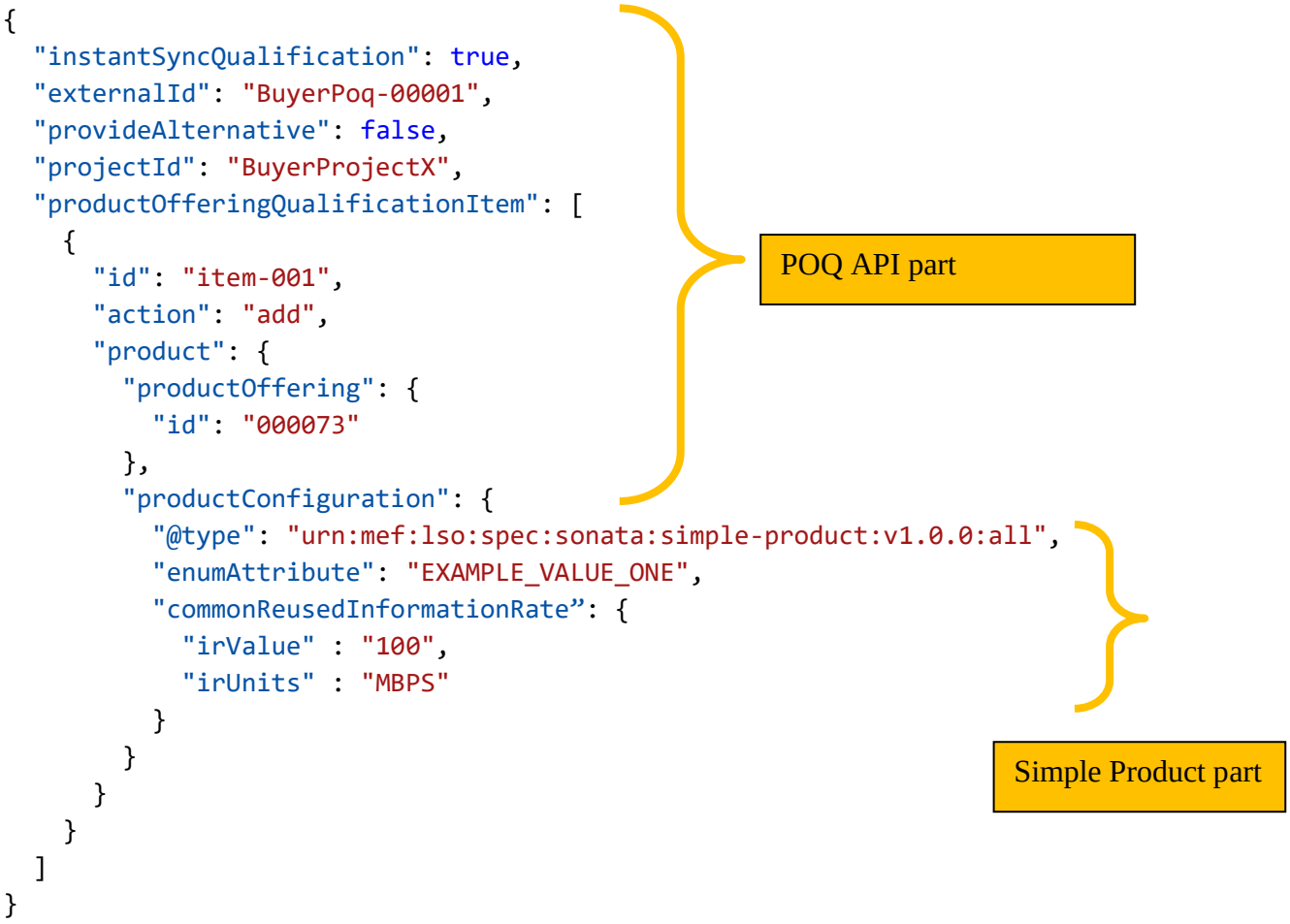



**Figure 8 – The extension pattern**

Payload specifications are provided as JSON schemas without the `MEFProductConfiguration` context. Payload-specific attributes are introduced via the `MEFProductRefOrValue` (defined by the Buyer). This type has the `productConfiguration` attribute of type `MEFProductConfiguration` which is used as an extension point for product-specific attributes. The example result of combining an envelope with a payload in a request JSON may look like below. Please refer to API Developer Guides (like MEF W87 [7]) or Schema Guides (like MEF W106 [8]) available at the SDK release for extensive explanation and examples.




## 5.2 Format

Payload specifications must be provided in the format of a JSON schema based on JSON schema draft 7 [1] and encoded in YAML for consistency with MEF-Standard schemas. Please see attached examples for reference.

Payload specifications must contain the ”$id” property so that it can be uniquely identified. The syntax of this field is explained in section 5.5.

```
"$id": urn:mef:lso:spec:sonata:simple-product:v1.0.0:all
```

The details on how to deliver the files are described in section 5.10.

## 5.3 Naming Conventions

The type names must follow the UpperCamelCase naming convention (e.g. `SelfDefinedEnumeration, InformationRate`)

The attribute names must follow the lowerCamelCase naming convention (e.g. `enumAttribute`, `commonReusedType`)

*Note*: In the payload schema no place explicitly defines the name of the root type. In the case of MEF Standard payloads, the name of the root type while binding (section 5.12) is decoded from the URN which contains the name of the product in its 6th part. (refer to section 5.5 for details of the "$id" structure). In case of Non-MEF Payloads, the product name will be taken from the file name which contains the root schema.

## 5.4 API flavors

The APIs that use the product payloads are LSO Cantata and LSO Sonata. They define a common end-to-end flow, built from several functional steps:

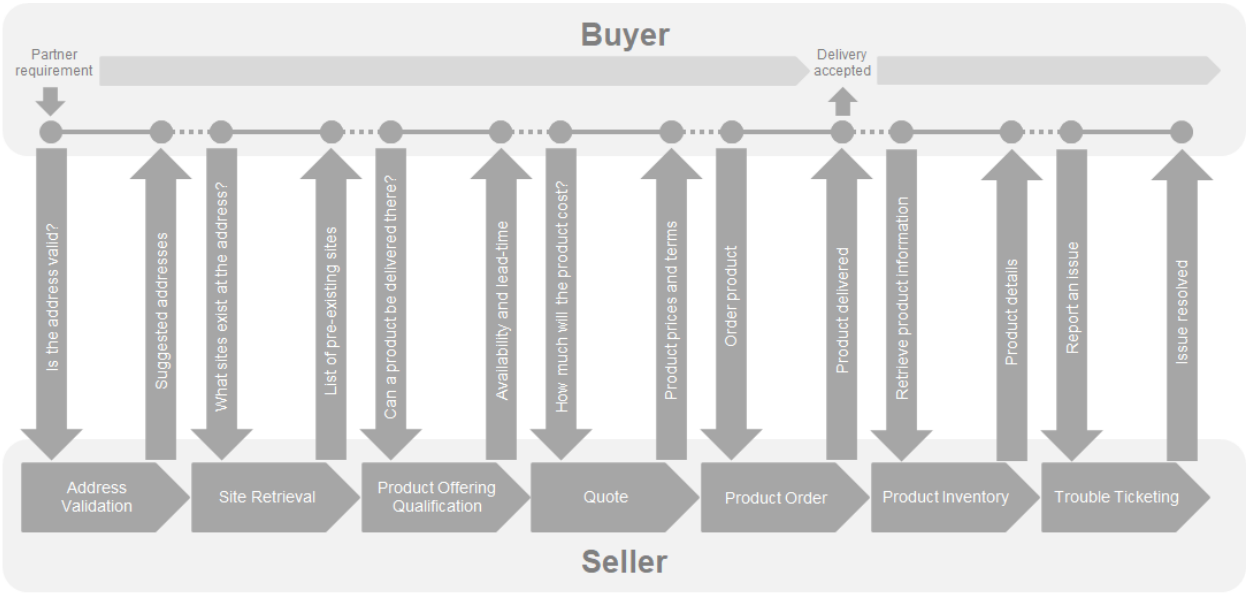



**Figure 9 – LSO Cantata and LSO Sonata End-to-End Function Flow**

- 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 the 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 for managing issues and situations that are not part of the normal operations of the Product provided by the Seller.

In the list above, the POQ, Quote, Order, and Inventory LSO APIs are product-oriented (i.e. carry a product payload). Each of these steps may or may not have different requirements on which attributes may or must be provided. If the requirements are consistent among the steps, one product schema can be provided. If the requirements differ between the steps –separate product schemas must be provided for each of the APIs.

Assuming POQ is the first step and Inventory is the last step, each next step:

- Can add new attributes

- Can mark more attributes as required

- Cannot modify attribute definition

It is a common approach in MEF Standard product schemas that there is no API flavor differentiation and that all attributes are optional. It is intended to be a subject of the onboarding process between the Buyer and the Seller to agree on which attributes will be used and/or mandated.

## 5.5 “$id”

The value must uniquely identify a payload specification. It should contain information about the payload type name, version, and use context (poq, quote, order, inventory). It must have the URI format, as specified in section 8.2 of [1].

Below you can find a description of MEF approach to ‘id’ governance for MEF-Standard payloads (for informational purposes).

To ensure uniqueness, MEF uses its registered URN space and specifies several requirements on its structure. The details are described on: MEF Assigned Names and Numbers, together with information on how to apply for one. The structure is presented below:

| No. | Meaning | String | Comments |
|---|---|---|---|
| 1 | Scheme | "urn" | |
| 2 | Authority | "mef" | |
| 3 | Namespaces Specific String Root | "lso" | MEF Project |
| 4 | Branch under "lso" | "spec" | This is the only branch currently defined. |
| 5 | LSO Interface Reference Point | <irp> | One of "cantata", "sonata", "cantata-sonata", "allegro", "interlude", "interlude-allegro", "legato", "presto", "adagio". |
| 6 | Product or Service Name | e.g., "access-eline:" | Other examples are "epl:", "subscriber-ethernet-uni:", "evp-lan:", "internet-access:", etc.<br><br>This field uses kebab-case |
| 7 | Version | e.g. "v1.0.0" | The semantic version number of the corresponding schema. |
| 8 | API Function | e.g. "poq" | Other examples are "quote", "order", "inventory". If a single schema is used for all functions, the API Function should be "all". |

**Table 1 – MEF LSO URN format**

The urns in this document, although being compliant with MEF urn standard syntax, are not registered officially in MEF's namespace. They are only to serve as examples within this document.

The urn of the Simple Product example is:

- `"$id": urn:mef:lso:spec:sonata:simple-product:v1.0.0:all`
    - `sonata` - It is intended to be used in the context of the LSO Sonata IRP. This is just for the sake of the example. It is the question if the product is to be used between Service Providers of between Service Provider and a Customer.
    - `simple-product` - the product name is "`simple-product`".
    - `v1.0.0 -` its version is 1.0.0
    - `all -` this one flavor will serve all LSO Sonata APIs. This is decided for the sake of example as Simple Product has only a few attributes.

This is the urn of the Complex Product:

- `"$id": urn:mef:lso:spec:cantata-sonata:complex-product:v1.0.0:poq`
    - `cantata-sonata` - it can be used in the context of both LSO Cantata and LSO Sonata (arbitrary example decision)
    - `complex-product -` the product name is "`complex-product`".

- `v1.0.0` - its version is 1.0.0
- `poq` - this particular flavor presented in the example file will serve only in the context of the POQ function. The remaining functions must use other flavors (also provided in the example files) which must be provided in the product specification.

## 5.6 MEF Common Model Use

During the development of MEF Standardized Payloads, repetitive parts are extracted into common, reusable schemas that are referenced by payload specifications. Depending on the scope of reusability those can be general, technology, or product common schemas. They should be reused if a payload specification attempts to model a type that is already available at MEF common types.

If the MEF-common type or dictionary requires some adaptation (e.g. "`irUnits`" in "`InformationRate`" should be narrowed to contain only "`MBPS`" and "`GBPS`" instead of a full list), the type can be replaced with a specialized one with the same name and some specific suffix added. (e.g. "`InformationRate` " => "`InformationRate_MBPS_GBPS`").

In Simple Product, the example of reuse of MEF Standardized model is the "`commonReusedInformationRate`". It is a 2-attribute ("`irValue`" and "`irUnits`") structure used to specify the information rate. It should be reused whenever this kind of information is modeled. The specification refers to the "`InformationRate`" type by pointing to the yaml file containing several reusable types: "`utilityClassesAndTypes.yaml`" is available in the "../`common`/" directory of attached example package.

The definition of the attribute looks as follows:

```
  commonReusedInformationRate:
    description: Demonstrates a use of a type referred from MEF common types'
definitions
    type: object
    $ref: "../common/utilityClassesAndTypes.yaml#/definitions/InformationRate"
```

The MEF common model schemas can be found in several places. There is a dedicated MEF Wiki page that provides updated information on respective artifacts or repositories.

*Note:* The method how MEF will provide the access to it's common model is undergoing a further study and is a subject to change.

## 5.7 Extracting Reusable Model

It is also highly recommended that if a newly prepared Non-MEF schema includes multiple usages of a newly defined type, then the type definition should be extracted to a common part and be referenced from multiple schemas to avoid duplication.

## 5.8 Internal Product/Service Dependencies

There are situations where attributes are dependent on other attributes. In such cases, the payload schema specification alone is not sufficient to ensure validation of the related business level requirements.

For example, conditional requirement, identifier validity, or more complex validation or mapping rules, like ("linkAggregation" attribute description from MEF 106, section 13.10 [8]):):

*Property Name: linkAggregation - Type: enum*

*Description: If the ENNI is composed of multiple physical links this Service Attribute indicates how they are combined using Link Aggregation.*

*Allowed values: "NONE", "2_LINK_ACTIVE_STANDBY", "ALL_ACTIVE", "OTHER"*

*Usage: poq: Not Included quote: Not Included order: Not Included inventory: Optional*

*Validation Notes: This needs to be validated against the x= cardinalityEnniCommon.listOfPhysicalLinks. If x=1 this must be "NONE". If x=2 this can be any of the allowed values other than "NONE". If x>2, this must be "ALL_ACTIVE" or "OTHER"*

This kind of information must be specified precisely in the schema's description. It is also recommended to have the additional requirements described in the accompanying documentation.

## 5.9 Payload-Related Envelope Requirements

The LSO API envelope model is used to specify the relations between the carried product and other products, as described in 5.9.1, and places, as described in section 5.9.2. This information should not be specified in the LSO Payload schemas.

### 5.9.1 Relations to other products

References between products are specified with the use of envelope attributes that are not product-specific (e.g. "`productRelationship`"). To distinguish the possible roles of these relations (e.g. startEndPoint, endEndPoint), the attribute "`relationshipType`" is used. If the payload specification defines any relation to other products (or services) the permitted values of the "`relationshipType`" attribute must be specified. An example of such information, taken from Access E-Line product specification (MEF W106, chapter 11, [8]) is presented in Table 2.

- Product Relationship Type – the allowed values of the "`productRelationship.relationshipType`" attribute that is used to distinguish different types of relations with other products.

- Install/Change – if the relationship is mandatory or not in Create and Modify operations.

- Product Specification – What type of other product is to be referred to by this relationship type

- Multiple Allowed at POQ and Quote – POQ and Quote steps allow pointing to multiple products with the same relation type to serve the use cases of site candidates. This may or may not be leveraged by the payload's use cases.

**Table 2 – Product Relationship Type - example**

| | Product Relationship Type | Install | Change | Product Specification | Multiple Allowed at POQ and Quote? |
|---|---|---|---|---|---|
| **Access E-Line** | CONNECTS_TO_ENNI | Mandatory | Mandatory | ENNI | Yes |
| **Access E-Line** | CONNECTS_TO_UNI | Mandatory | Mandatory | UNI | No |

Information should be provided on whether modification of already existing relationships is allowed. Technically the question is whether a different value is allowed to be specified in a change request then the value that was specified in the original install request Such a use case can potentially mean moving the connection endpoint from one place to another. This usually is forbidden so the product must be deleted first and then created again with a new configuration to support such a use case.

There are two ways to refer to the products depending on if they already exist in the inventory. Specified relationship roles apply to both of them in the same way.

`MefProductRefOrValue.productRelationship.relationshipType` – Product relationship is used when a given product has a relation to a product already existing in the inventory.

`…itemRelationship.relationshipType` – depending on the API it will be `qualificationItemRelationship`, `quoteItemRelationship`, or `orderItemRelationship`. Item relationship is used when the dependency is not pointing to an existing product but to one which is a subject of the same request and is described by a sibling item (poq, quote, or order).

### 5.9.2 Place relationship

`MefProductRefOrValue.place` – when the payload model requires reference to an Address or Site it should use this reference attribute. This is because there are two APIs dedicated to using cases of Address Validation (MEF W121, [9]) or Site Retrieval (MEF W122, [10]) that serve the purpose of acquiring proper ids from the Seller. Additionally, the "`place`" has a mandatory attribute "`role`" which has the same function as the "`relationshipType`" described in the section above, and the permitted values (if any) must be specified by the payload documentation. Again, information should be added on whether it is mandatory to provide the relationship per action type, together with a statement if an update is possible. It is recommended to provide it in a form of a table like in the example below (MEF 106, section 13.10 [8]).

- Place Relationship Role – the permitted values of the "`placeRelationship.role`" attribute that is used to distinguish different types of relations with other products.
- Install/Change – if the relationship is mandatory or not in Create and Modify operations.

**Table 3 – Place Relationship Role – example**

| | Place Relationship Role | Install | Change |
|---|---|---|---|
| **UNI** | INSTALL_LOCATION | Mandatory | Mandatory |

…`item.relatedContactInformation.role` – Payload specification may also mandate providing contact information of some role. It is introduced via the "`relatedContactInformation`" attribute at the item level (poq, quote, or order).

## 5.10 Packaging

The payload package must be delivered as a zip of all relevant schemas in the proper directory structure. It should only contain schemas that are referenced by the actual payload schemas. The required directory structure is as follows:

- The Payload type: (product|service)
  - Technology: (carrierEthernet|ip|sdWan| etc…)
    - Optionally: (common)
    - Optionally directory per product

Figures below present example delivery packages of the example products.

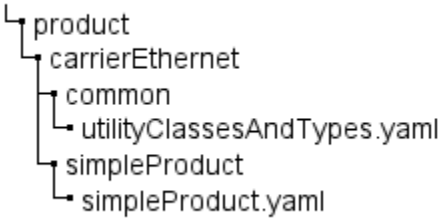



**Figure 10 – Simple product package**

The Simple Product has only one flavor for all API functions (POQ, Quote, Order, Inventory) so it is delivered as a single file: "/`product/carrierEthernet/simpleProduct/simpleProduct.yaml`". It contains the "`all`" suffix in the "`$id`" to reflect that. It refers only the "`InformationRate`" common type, which is available in

"/product/carrierEthernet/common/utilityClassesAndTypes.yaml". Out of all MEF common schemas available, only this one is included in the package. While the common files may be selectively chosen to be included in the package, their content must not be changed, even if not all types are referenced.

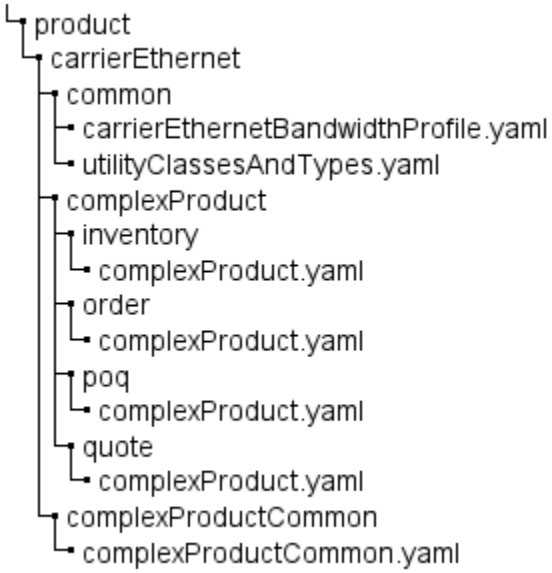



**Figure 11 – Complex Product package**

The Complex Product package has a more complex structure. Firstly, it comes in different flavors per API function, thus the "/complex/" folder is further split into "/inventory/", "/order/", "/poq/", and "/quote/". Each of them contains the respective schema for a given function. Additionally, it introduces the complexProductCommon.yaml which provides schemas that are referred to by others (see the snippet below).

```
(inventory/complexProduct.yaml)
…
allOf:
  - $ref:
"../../complexProductCommon/complexProductCommon.yaml#/definitions/ComplexProduct
Common"
  - type: object
    required:
      - ceVlanIdPreservation
      - cTagPcpPreservation
      - maximumFrameSize
      - listOfClassOfServiceNames
      - startEp
      - endEp
```

```
    properties:
      startEp:
        description:
          Start EndPoint representation
        type: object
        $ref: "#/definitions/ComplexProductEndPoint"
      endEp:
        description:
          End EndPoint representation
        type: object
        $ref: "#/definitions/ComplexProductEndPoint"
```

This in particular contains the `ComplexProductCommon` and `ComplexProductEndPointCommon` types. This mirrors the pattern used (e.g. the MEF W106 Access E-Line product specification ([8]). This is the case when the payload specification varies only by the list of required attributes. To avoid duplication the "…`Common`" types define all attributes as not required. The final specification of "`ComplexProduct`" per function references its corresponding Common type with the "`allOf`" which allows adding the "`required`" statement and list which attributes are considered mandatory in the given context.

The "`startEp`" and "`endEp`" are the only attributes that are defined by the end schemas. This is done to introduce the "`ComplexProductEndPoint`" type that extends the "`ComplexProductEndPointCommon`" to define the desired required list. In the case that the "`ComplexProductEndPoint`" type doesn't differ between the functions, it should be fully defined in the common schemas.

### 5.11 Static and dynamic binding

As mentioned in section 5.1 there are two building blocks of MEF LSO APIs: the functional product or service agnostic APIs (envelope), and the product or service-specific payload. They are separate artifacts and the Seller and the Buyer must communicate on what schemas are supported by the API. This can be achieved in two ways: **Dynamic binding** or **Static binding**.

In the **Dynamic Binding** approach, the Seller documents (using OAS) only the envelope part of the API. Looking at the documented endpoint specification, the Buyer is not able to know which products or services the Seller supports. This information must be shared during the onboarding process or with the use of the Product Catalog. This approach allows for the easy and dynamic addition of support of the new schema without system redeployment.

An example implementation of the dynamic binding approach can be found in the Example-LSO-Dynamic-Binding-Implementation GitHub repository (available only to MEF members).

In the **Static Binding** approach, the endpoint specification exposed by the Seller is integrated in advance with the data model of supported product/service specifications. Looking at the API OAS documentation the Buyer can tell which products/services are supported. In this case adding support for new payload type requires a change in the API specification and its redeployment. Payload specifications are in an inheritance relationship with "`MEFProductConfiguration`"

as described in the API specification. The "`@type`" attribute is a discriminator used to map the payload specification ids to corresponding resources of the API specification.

A static binding of example product specifications with POQ API would look like presented in the snippet below. This is an extract of the "`productOfferingQualificationManagement.api.yaml`" containing only the "`MEFProductConfiguration`" and root example Product types. An example set of a bound set of APIs is attached in Appendix A.

```
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:simple-product:v1.0.0:all:
'#/components/schemas/SimpleProduct'
      urn:mef:lso:spec:cantata-sonata:complex-product:v1.0.0:poq:
'#/components/schemas/ComplexProduct'
    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
SimpleProduct_v1.0.0:
  allOf:
    - $ref: '#/components/schemas/MEFProductConfiguration'
    - description:
        This simple example specification aims to demonstrate the basic technical
        considerations for preparing a valid MEF LSO Payload specification.
        # ...
ComplexProductPoq_v1.0.0:
  allOf:
    - $ref: '#/components/schemas/MEFProductConfiguration'
    - description:
        This example specification aims to demonstrate advanced technical
        considerations for preparing a valid MEF LSO Payload specification (e.g.
        multi-API flavors pattern)
        # ...
```

## 5.12 Binding Tool

LSO SDK releases provide a non-normative artifact of all product-oriented API statically bound with all compatible MEF Standardized product or service specifications (where applicable). The purpose of this is to make the implementation faster. However, since over time the number of available standard product and service specifications will grow, this will become too extensive and unusable for service providers offering just a subset of products. There is a need to cherry-pick only the required payloads and bind them selectively with the relevant LSO APIs.

A manual binding of envelopes and payloads can be very time-consuming and error-prone. MEF uses an open-source tool available on GitHub: SonataBindingTool. It is a Java-based command-line tool. It allows both bulk and selective binding.

With a single command, a developer can statically bind several payload specifications into one functional API schema. For example, a command that would bind examples payloads into the POQ API:

```
java -jar .\blender-1.6.jar blend
-i C:\pathTo_productOfferingQualificationManagement.api.yaml
-p C:\pathTo\carrierEthernet\complexProduct\poq\complexProduct.yaml
-p C:\PathTo\carrierEthernet\simpleProduct\simpleProduct.yaml --sorted
```

A staticly bound pack of APIs with example payloads (as would be included in an LSO SDK) can be found in Appendix A.

The binding tool is also used to perform the previously mentioned Binding Test in Sections 3.1 and 3.2. It verifies some basic technical requirements like:

- basic schema syntax validation
- directory structure
- references validity

# 6 Non-MEF LSO Payload Documentation Requirements

Documentation is a requirement for MEF-Endorsed LSO Payloads to be included in the LSO Marketplace. This section explains the documentation requirements and development guidelines.

## 6.1 Document Format

The document must be provided using the MEF template that is available on request from LSO Developer Community Manager. It is recommended that documentation also be included within the documented schema in the descriptions.

## 6.2 Business description

The document must include a business-oriented description of the product or service and its schema. The description should include the following information with both text and accompanying diagrams where possible. If the specification is based on a standard (MEF or other SDO), respective sections can be provided via reference.

### 6.2.1 Product/Service Description

An explanation must be provided of the externally visible behavior of the product or service, where and how it is used, and by who. If any market information is available, that should be included. Sufficient information needs to be provided to enable a new third party to implement or use the product or service correctly.

### 6.2.2 Usage Requirements and Restrictions

If there are any business requirements of users of the payload schema, such as license agreements, license fees, IPR (Intellectual Property Rights), etc. that should be taken into account, these must be clearly described.

### 6.2.3 Use Cases

Use cases describing various real-life typical appliances must be provided, including the typical interaction between the entities using and providing the product/service. They must describe various Internal Product/Service Dependencies

If there are any internal dependencies between objects and attributes, as described in section 5.8, they must also be described in the documentation.

## 6.3 Payload-Related Envelope Requirements

Section 5.9 describes the envelope-related requirements that must be taken into consideration and documented in the API description part. If applicable they must be also provided in the documentation in a more descriptive way.

## 6.4 Buyer-Seller Onboarding Information.

MEF LSO APIs aim to maximize interoperability. The standardization process aims to leave as few decisions or interpretations to the implementer of the API as possible. The documentation must explicitly describe all issues that need to be agreed upon on between the Buyer and the Seller that cannot be understood directly from the Non-MEF Payload schema. The process of this bilateral agreement is often referred to as onboarding.

## 6.5 Order delivery lifecycle milestones

The LSO Order API supports notification of achievement of ordering milestones. These milestones are strictly related to the specifics of the product's lifecycle. They may be defined by the product specification if needed. Table 4 shows an example milestone list for Access E-Line, as provided in [8] in chapter 8.

**Table 4 – Order Milestones for Access-E-Line**

| Milestone Value | Description | Applies To |
|---|---|---|
| SITE_SURVEY_SCHEDULED | Site Survey Scheduled | UNI |
| SITE_SURVEY_COMPLETE | Site Survey Complete | UNI |
| PLANNING_COMPLETE | Planning Complete | UNI, OVC |
| FIRM_DELIVERY_DATE_PROVIDED | Firm Delivery Date Provided | UNI, OVC |
| AWAITING_MUNICIPAL_APPROVAL | Awaiting Municipal Approval | UNI |
| MUNICIPAL_APPROVAL_GRANTED | Municipal Approval Granted | UNI |
| AWAITING_LANDLORD_APPROVAL | Awaiting Landlord Approval | UNI |
| LANDLORD_APPROVAL_GRANTED | Landlord Approval Granted | UNI |
| CONSTRUCTION_STARTED | Construction Started | UNI |
| CONSTRUCTION_COMPLETED | Construction Completed | UNI |
| AWAITING_ACCESS | Awaiting Site Access Permission (for end-to-end test) | UNI, OVC |
| ACCESS_DENIED | Site Access Denied (for end-to-end test) | UNI, OVC |
| AWAITING_WIRING | Awaiting Installation of Inside Wiring by Landlord | UNI |
| WIRING_COMPLETE | Installation of Inside Wiring by Landlord Complete | UNI |
| EQUIPMENT_DISPATCHED | Equipment Dispatched | UNI |
| EQUIPMENT_DELIVERED | Equipment Delivered | UNI |
| EQUIPMENT_INSTALLED | Equipment Installed | UNI |
| E2E_TESTING_SCHEDULED | End-to-End Testing Scheduled | OVC |
| E2E_TESTING_COMPLETED | End-to-End Testing Completed | OVC |
| E2E_TESTING_FAILED | End-to-End Testing Failed | OVC |

## 6.6 Data Model

The documentation should contain as detailed a specification as possible of the data model with all the product or service objects and attributes. The specification should contain:

- Attribute name – the name of the attribute in a human-readable form (e.g. *Committed Information Rate*)
- Attribute JSON name – the name of the attribute as provided in the JSON schema (e.g. *cir*)
- Type – the type of the attribute

- Description – explanation of meaning, semantics, all the requirements, and rules that apply
- Allowed values – to specify the dictionary values, ranges, format, syntax, class types, etc.
- Usage – statement if the attribute is required or optional per poq, quote, order, and inventory APIs

## 6.7 Examples in different configurations and contexts

Comprehensive examples of how to use the payload in different contexts and versions are very useful for developers and should be included in the LSO Payload documentation as an appendix. Please see MEF W106, Appendix A ([8]) for reference.

- A list of use cases
- A list of some typical real-life payload examples
- Diagrams supporting the examples
- A set of examples guiding the usage of the payload through different envelope APIs
- Examples of different actions – add, modify (based on real-life use cases), delete
- Description of some corner cases or other problematic situations (e.g. the need to disconnect and create a new connection instead of modifying its termination point)

The full examples may be extensive, and often going through the variations may result in duplication of large parts of the API request example payloads. Therefore, only the first few examples in the document need to use a full request to provide an explanation, whereas the remaining examples can highlight only the use case specifics. It is recommended that all full example requests should be provided as an external set of files, preferably in the form of Postman collections.

# 7 References


[1] *JSON Schema specification*, https://datatracker.ietf.org/doc/html/draft-handrews-json-schema-01

[2] MEF 26.2, *External Network Network Interfaces (ENNI) and Operator Service Attributes*, August 2016

[3] MEF 51.1, *Operator Ethernet Service Definitions*, December 2018

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

[5] MEF 70.1, *SD-WAN Service Attributes and Service Framework*, November 2021

[6] MEF 79, *Address, Service Site, and Product Offering Qualification Management Requirements and Use Cases*, November 2019

[7] MEF W87, *Product Offering Qualification Management API Developer Guide*, January 2022

[8] MEF W106, *LSO Sonata Product Specification - Access E-Line - Schema Guide*, June 2021

[9] MEF W121, *LSO Cantata and LSO Sonata Address Management API - Developer Guide*, January 2022

[10] MEF W122, *LSO Cantata and LSO Sonata Site Management API - Developer Guide*, January 2022

[11] RFC 8141, *Uniform Resource Names (URNs)*, Peter Saint-Andre and Dr. John C. Klensin, April 2017. Copyright © 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

# Appendix A LSO Payload Examples

This appendix contains examples of a simple product and of a complex product in the form of payload specifications.

- Simple Product:

simpleProduct.zip

- Complex Product:

complexProduct.zip

- Static binding:

generated.zip

# Appendix B Proposal Form for Non-MEF LSO Product Payload

If a MEF member is interested in having its product or service schema posted in the LSO Marketplace as a Non-MEF LSO Payload (MEF-Endorsed or Partner-Specific), it can submit a proposal to MEF via the LSO Developer Community Manager (community_manager@mef.net). The short proposal enables the MEF membership to check that the proposed payload schema would not conflict with any ongoing MEF work or otherwise confuse the market. The proposal must include:

- The name of the MEF member making the proposal and full contact details (Note that only MEF member companies can propose a non-MEF LSO Payload. However, a non-MEF member can partner with a MEF member to make a proposal, and the involvement of a non-MEF member in the proposal should be noted)
- If this is a product or service specification
- LSO APIs with which the payload would be used
- Preferred type of MEF Payload (MEF-Endorsed or Partner-Specific)
- A short business description of the product or service
- Payload schema (an initial version is acceptable)

# Appendix C

## C.1 MEF-Standardized Payloads

MEF standardized payloads can be found:

- within the MEF LSO SDK release on GitHub. The list of all MEF's GitHub SDKs is presented on this LSO Marketplace subpage. Taking Sonata SDK Billie release (public), for example, product schemas can be found in the `\productSchema` directory. Supporting documentation can be found in the `\documentation\productSchema` directory. A particular release contains only the payloads applicable for the APIs within. Alike the APIs, the payload schemas included in the SDKs might not necessarily be a published standard version. A work in progress or draft standard versions are also published. Note: In some of the releases the documentation is available only in the extended version of the SDK release, available only MEF-members.

- LSO Marketplace

This type is provided as a MEF published standard which is delivered by MEF community project and brings broad MEF knowledge and industry consensus. This process involves the following steps:

- Specification of Service Attributes and Service Definition:
    - Definition and detailed explanation of the service model and its attributes, functionalities, configuration rules, dependencies, etc. provided in a form of a document.
- Specification of Service schema
    - service's data model provided in a form of a JSON schema file and supporting documentation
    - based on Service Attributes and Definition
    - applicable for service-oriented APIs: Legato, Allegro, and Interlude
- Specification of Product Schema
    - product's data model provided in a form of a JSON schema file and supporting documentation
    - based on the Service Attributes and Definition data model
    - applicable for product-oriented APIs: Cantata and Sonata

Depending on the complexity (or other different concerns) the first two steps are delivered by a single document (e.g. MEF 70.1 for SD-WAN [5]) or separate documents (e.g. MEF 26.2 [1] and MEF 51.1 [3] for Carrier Ethernet). Service Model and Product Model specifications come as separate standards. Figure 12 presents the standardization steps together with examples of standards that delivered them.


**Figure 12 – MEF-Standard LSO Payload development process**