

LSO Payload HandbookJanuary 2022

Working Draft

LSO Payload Handbook

January 2022

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

Disclaimer

© MEF Forum 2022. All Rights Reserved.

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

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

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

1. 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
2. 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
3. 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

[1 Terminology and Abbreviations 1](#_Toc92780756)

[2 Introduction 2](#_Toc92780757)

[2.1 MEF-Standardized LSO Payloads 2](#_Toc92780758)

[2.2 Non-MEF LSO Payloads 3](#_Toc92780759)

[2.2.1 MEF-Endorsed LSO Payloads 4](#_Toc92780760)

[2.2.2 Partner-Specific LSO Payloads 4](#_Toc92780761)

[2.3 Choosing an LSO Payload Goal 4](#_Toc92780762)

[3 Governance 5](#_Toc92780763)

[3.1 MEF-Endorsed LSO Payloads 5](#_Toc92780764)

[3.2 Partner-Specific Payloads 6](#_Toc92780765)

[3.3 Comparison of LSO Payload Development Processes 7](#_Toc92780766)

[4 Non-MEF LSO Payload Technical Requirements 7](#_Toc92780767)

[4.1 Integration of payload specifications into the APIs 10](#_Toc92780768)

[4.2 Format 11](#_Toc92780769)

[4.3 Naming Conventions 11](#_Toc92780770)

[4.4 API flavors 12](#_Toc92780771)

[4.5 “$id” / URN structure 13](#_Toc92780772)

[4.6 Common Model Reuse 14](#_Toc92780773)

[4.7 Internal Product/Service Dependencies 15](#_Toc92780774)

[4.8 Payload-Related Envelope Requirements 16](#_Toc92780775)

[4.8.1 Relations to other products 16](#_Toc92780776)

[4.8.2 Place relationship 17](#_Toc92780777)

[4.9 Packaging 18](#_Toc92780778)

[4.10 Static and dynamic binding 20](#_Toc92780779)

[4.11 Blending Tool 21](#_Toc92780780)

[5 Non-MEF LSO Payload Documentation Requirements 23](#_Toc92780781)

[5.1 Document Format 23](#_Toc92780782)

[5.2 Business description 23](#_Toc92780783)

[5.2.1 Product/Service Description 23](#_Toc92780784)

[5.2.2 Usage Requirements and Restrictions 23](#_Toc92780785)

[5.2.3 Concepts and Information Model Objects 23](#_Toc92780786)

[5.2.4 Use Cases 23](#_Toc92780787)

[5.2.5 Internal Product/Service Dependencies 24](#_Toc92780788)

[5.3 Payload-Related Envelope Requirements 24](#_Toc92780789)

[5.4 Buyer-Seller Onboarding Information. 24](#_Toc92780790)

[5.5 Order delivery lifecycle milestones 24](#_Toc92780791)

[5.6 Data Model 25](#_Toc92780792)

[5.7 Examples in different configurations and contexts 25](#_Toc92780793)

[6 References 26](#_Toc92780794)

[Appendix A LSO Payload Examples 27](#_Toc92780795)

[Appendix B Proposal Form for Non-MEF LSO Product Payload 28](#_Toc92780796)

[Appendix C 29](#_Toc92780797)

[C.1 MEF-Standardized Payloads 29](#_Toc92780798)

[C.2 MEF-Standardized Payloads Roadmap 30](#_Toc92780799)

# List of Figures

[Figure 1 LSO Reference Architecture 2](#_Toc92780800)

[Figure 2 Categories of LSO Payloads 4](#_Toc92780801)

[Figure 3 MEF-Endorsed LSO Payload development processes 6](#_Toc92780802)

[Figure 4 Partner-Specific Payload development processes 7](#_Toc92780803)

[Figure 5 LSO Payload development processes 7](#_Toc92780804)

[Figure 6 Simple Product model 9](#_Toc92780805)

[Figure 7 Complex Product model 9](#_Toc92780806)

[Figure 8 The extension pattern 10](#_Toc92780807)

[Figure 9 LSO Cantata and LSO Sonata End-to-End Function Flow 12](#_Toc92780808)

[Figure 10 Common model schemas directory structure 15](#_Toc92780809)

[Figure 11 Simple product package 18](#_Toc92780810)

[Figure 12 Complex Product package 18](#_Toc92780811)

[Figure 13 MEF-Standard LSO Payload development process 30](#_Toc92780812)

# List of Tables

[Table 1 MEF LSO URN format 14](#_Toc92780813)

[Table 2 Product Relationship Types 17](#_Toc92780814)

[Table 3 Place Relationship Role 17](#_Toc92780815)

[Table 4 Order Milestones for Access-E-Line 24](#_Toc92780816)

# 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 |
| --- | --- | --- |
| **BUS** | An LSO Reference Architecture component approximately equivalent to the BSS layer. | MEF 55.1 [3] |
| **Envelope** | **LSO API Envelope**  Product or Service independent API (Function-specific information and Function-specific operations). (e.g. Product Order API) | This document |
| **IRP** | **Interface Reference Point**  The logical point of interaction between specific management entities | MEF 55.1 [3] |
| **LSO** | **Lifecycle Service Orchestration**  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, overall the network domains that require coordinated management and control to deliver the Service. | MEF 55.1 [3] |
| **MEF-Endorsed LSO Payload** | A non-MEF LSO Payload which has been both documented according to the requirements in this document and has been approved by the MEF membership for inclusion in the [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net) | This document |
| **MEF-Standardized LSO Payload** | A schema for use in LSO APIs that is based on MEF standards. | This document |
| **Non-MEF LSO Payload** | A schema for use in LSO APIs that is included in the [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net) but which is not based on MEF standards. | This document |
| **Partner-Specific LSO Payload** | A non-MEF 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](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net). | This document |
| **POQ** | **Product Offering Qualification** | MEF 79 [4] |
| **SDO** | **Standards Developing Organization** |  |
| **URI** | **Uniform Resource Identifier** a compact string of characters for identifying an abstract or physical resource classified as a locator, a name, or both | [RFC 2396](https://www.ietf.org/rfc/rfc2396.txt) [10] |
| **URN** | **Uniform Resource Names** A subset of URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable. | [RFC 2141](https://www.ietf.org/rfc/rfc2141.txt) [9]  [RFC 2396](https://www.ietf.org/rfc/rfc2396.txt) [10] |

# Introduction

MEF has standardized the use of APIs to automate business and operational interactions between Buyers and Sellers based on the LSO Reference Architecture (MEF 55.1 [3]). 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:

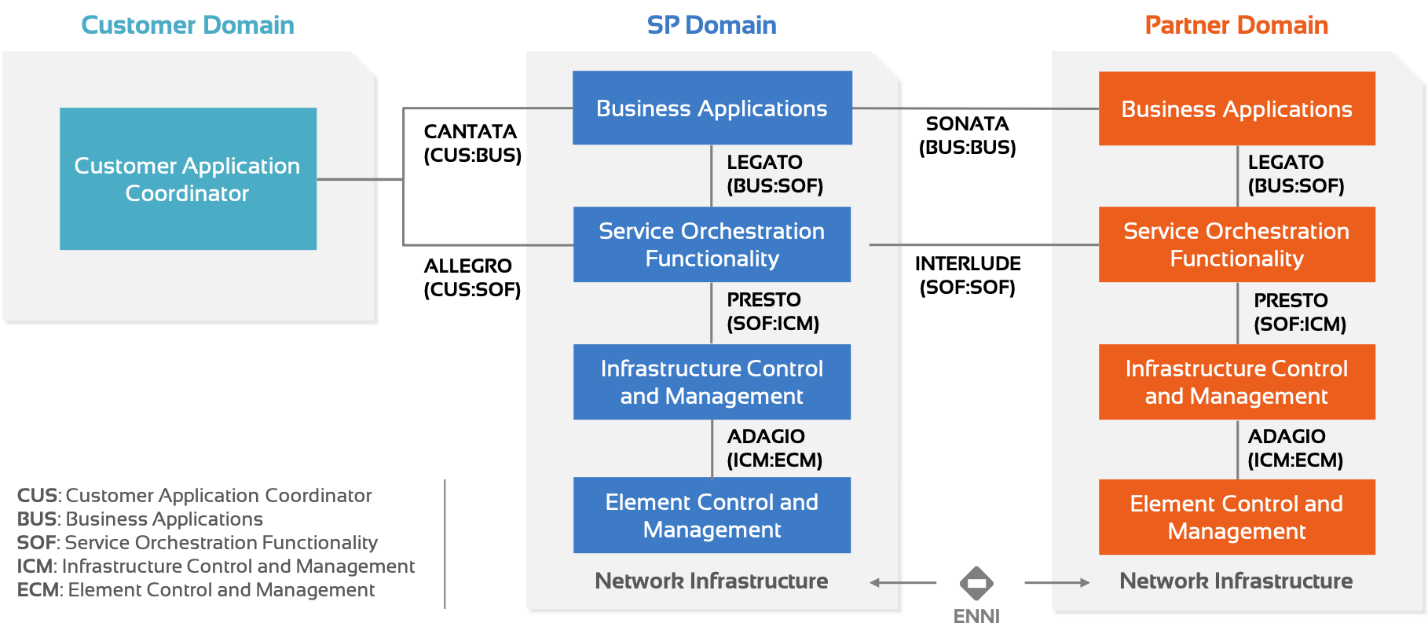


Figure LSO Reference Architecture

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 (“P**ayload”)**.

## MEF-Standardized LSO Payloads

Three of the IRPs interfacing with SOF (LSO Legato, LSO Allegro, LSO Interlude) support operational actions on a **service** detailed enough to order and configure a service in the network. Two of the IRPs interfacing with BUS (LSO Cantata and LSO Sonata) support the business actions on a **product** where a product is a business abstraction of a service that contains attributes only relevant for a business action. MEF standardizes both product and service specifications (e.g. Carrier Ethernet, Optical Transport, Internet Access, Subscriber IPVPN, SD-WAN, etc.). 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 Payload.

## Non-MEF LSO Payloads

However, by virtue of the polymorphic 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 marketing reasons. These implementers may want to use the [LSO Marketplace](lso.mef.net) to broaden the awareness of their Non-MED 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.) and 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.
* 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 used by developers of non-MEF LSO Payloads that want to have their schemas included in the MEF’s [LSO Marketplace](lso.mef.net).

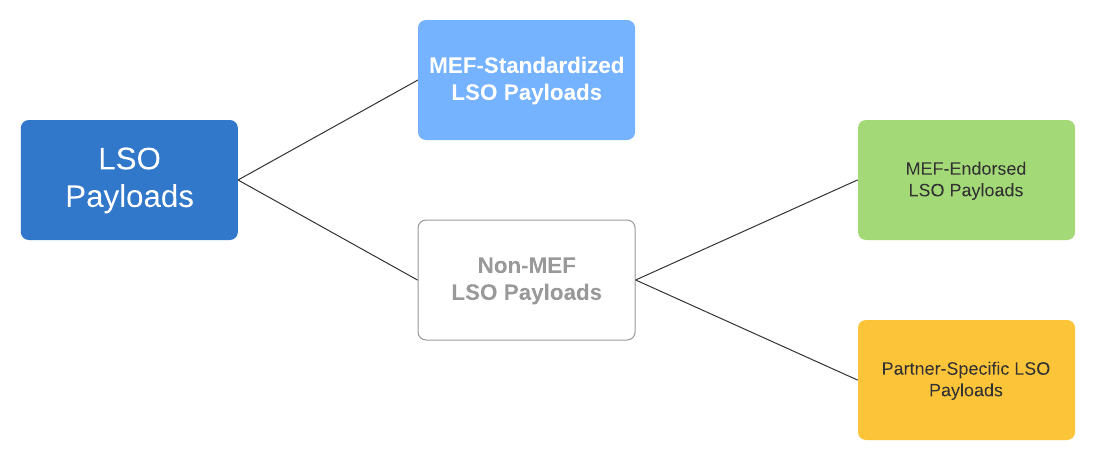


Figure Categories of LSO Payloads

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

### MEF-Endorsed LSO Payloads

These are schemas that meet the following requirements:

* Meet technical requirements described in Section 4
* Meet the documentation requirements described in Section 5
* Specifically approved for inclusion in the [LSO Marketplace](lso.mef.net) by the MEF membership as a ‘MEF-Endorsed LSO Payload’ described in Section 3.

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

### 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](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net) by the MEF membership as ‘Partner-Specific LSO Payload’ as described in Section 3.

## Choosing an LSO Payload Goal

Below are the general considerations and steps:

* Check the MEF Standardized payload roadmaps. 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.
* Fill the Assessment Form. 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 when the new payload would overlap with any other existing or developed one or is not breaking any MEF best practices.
* Read this Handbook. It provides technical 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 3.1 or 3.2 and build your schema(s).
* After getting the Assessment and Approval – your payload schemas will be published on the [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net).

# Governance

This section describes the process and rules for establishing Non-MEF LSO Payloads in the [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net). For reference purposes, information on the establishment of MEF-Standardized LSO Payloads in the LSO Marketplace is provided in Appendix C.

## 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 and documentation requirements
* MEF member submits a proposal (see Appendix B)
* MEF Commercial and Business Committee reviews the proposal and provides feedback to the MEF member
* If decided to continue, a two-week electronic ballot that is open to all MEF members is held to see if there is a majority for recommending to the MEF member to continue the proposal process.
* If the recommendation is to continue, the MEF will facilitate the testing of the proposed LSO Payload for successful blending in accordance with the technical requirements in Section 4.
* If the testing is successful, the MEF member completes the documentation according to Section 5.
* Once the documentation is completed, the proposal together with the documentation 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 documentation 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, then the LSO Payload is included in the LSO Marketplace as a MEF-Endorsed LSO Payload.
* Note that MEF-Endorsed LSO Payloads are available in the LSO Marketplace so long as all the following criteria are met:
  + 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 motion in the LSO Committee

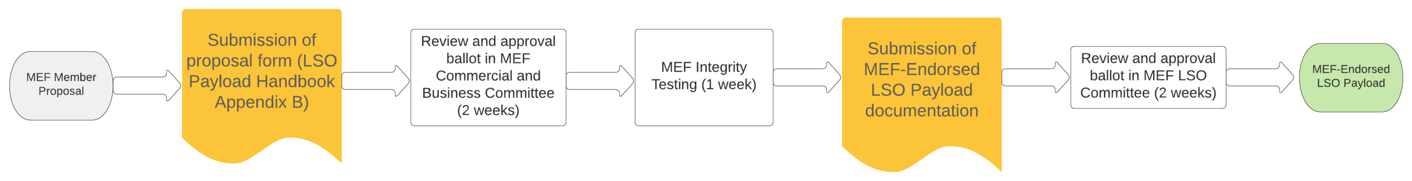


Figure MEF-Endorsed LSO Payload development processes

## Partner-Specific 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 requirements
* MEF member submits a proposal (see Appendix B)
* MEF Commercial and Business Committee reviews the proposal and provides feedback to the MEF member
* If decided to continue, a two-week electronic ballot that is open to all MEF members is held to see if there is a majority for recommending to the MEF member to continue the proposal process.
* If the recommendation is to continue, the MEF will facilitate the testing of the proposed LSO Payload for successful blending in accordance with the technical requirements in Section 6.
* If the testing is successful, the proposal together with the 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, then the LSO Payload is included in the LSO Marketplace as a Partner-Specific LSO Payload.
* Note that Partner-Specific LSO Payloads are available in the LSO Marketplace so long as all the following criteria are met:
  + 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 motion in the LSO Committee



Figure Partner-Specific Payload development processes

## 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.

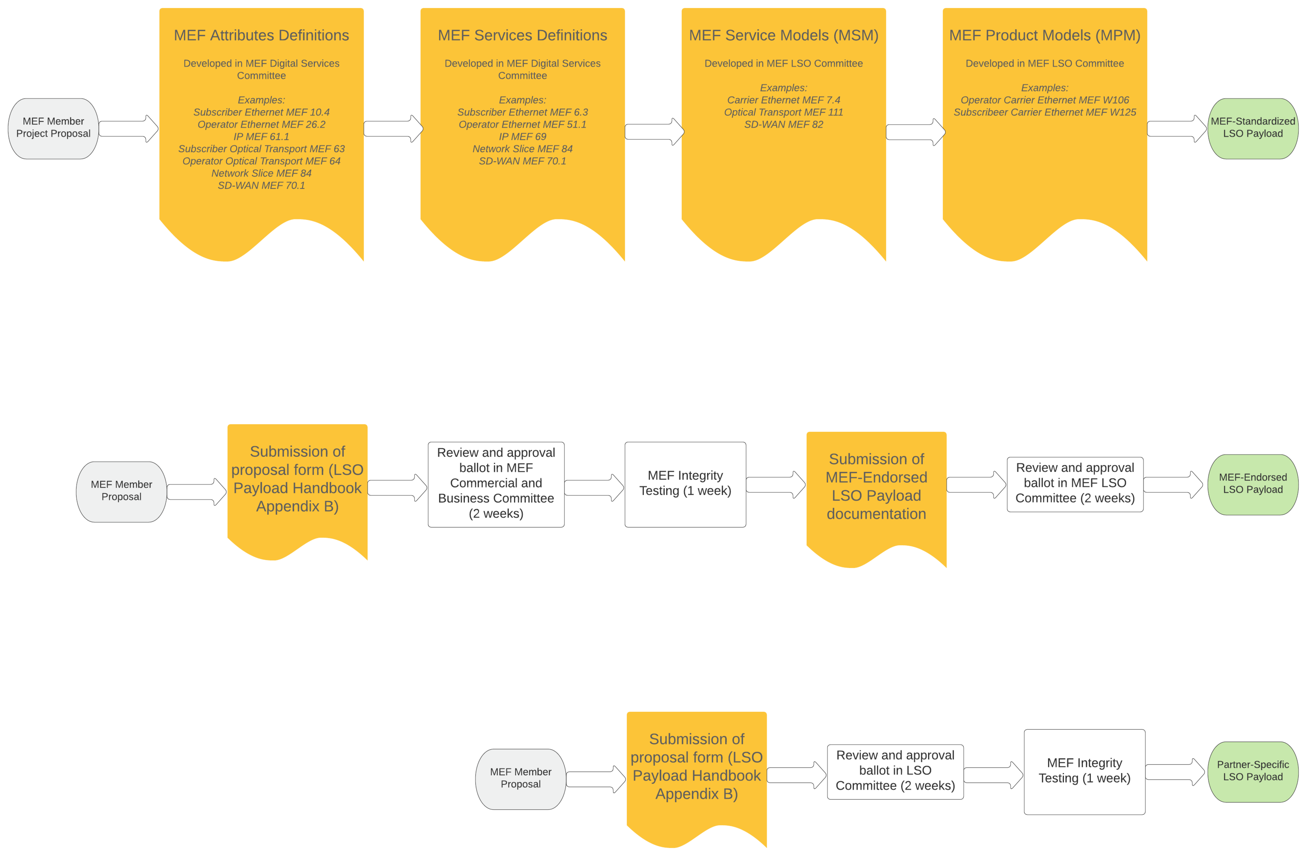


Figure LSO Payload development processes

# 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 and 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
* Blending tool

The technical requirements are described at two levels:

* Simple Product – to illustrate the basic concepts
* Complex Product – to illustrate advanced and real-life use cases.

*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 model examples.

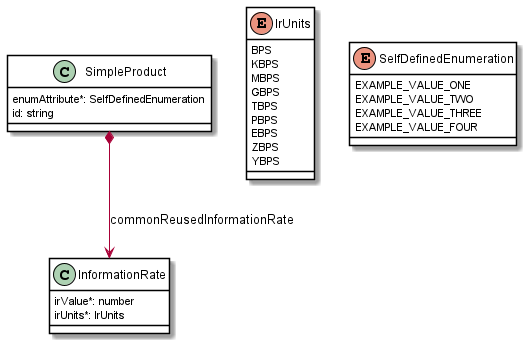


Figure Simple Product model

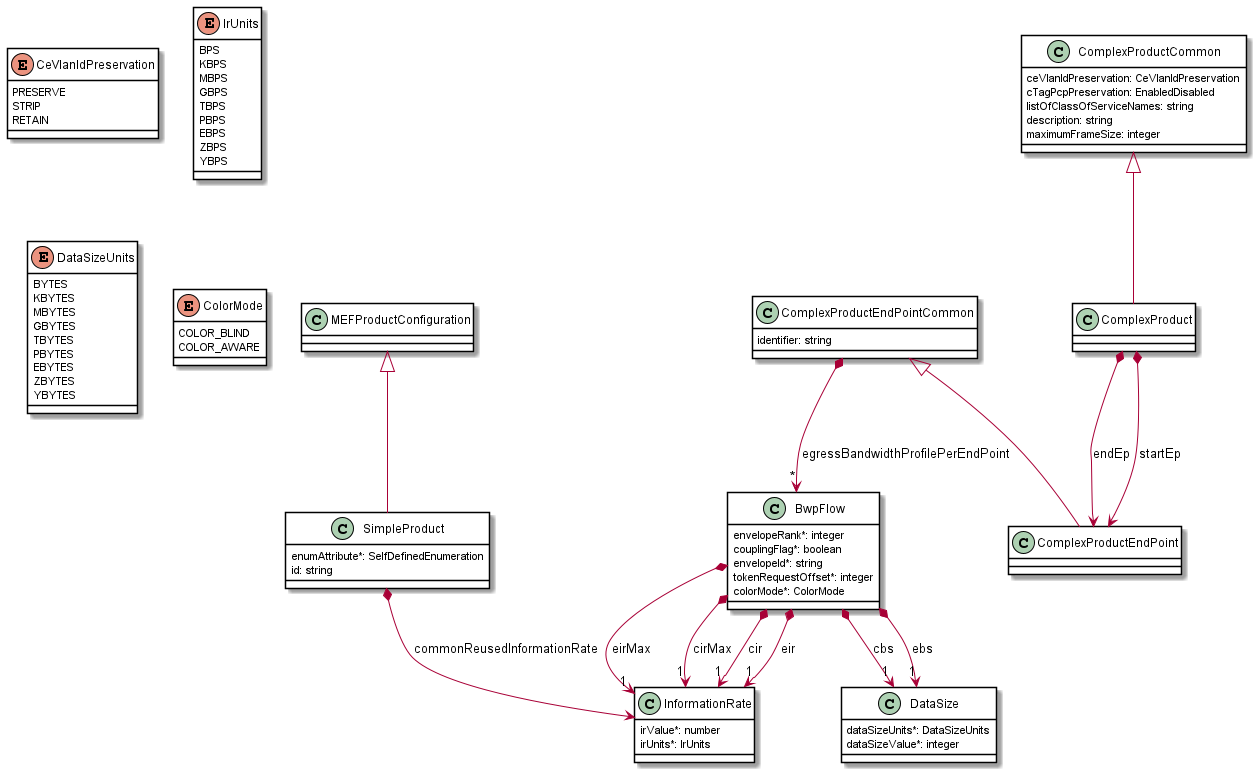


Figure Complex Product model

## Integration of payload specifications into the APIs

MEF LSO standard APIs are product/service-agnostic in the meaning that they serve as business interaction level 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.

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 provided as root payload schema (`$id`). In this document’s example schemas this will be:

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

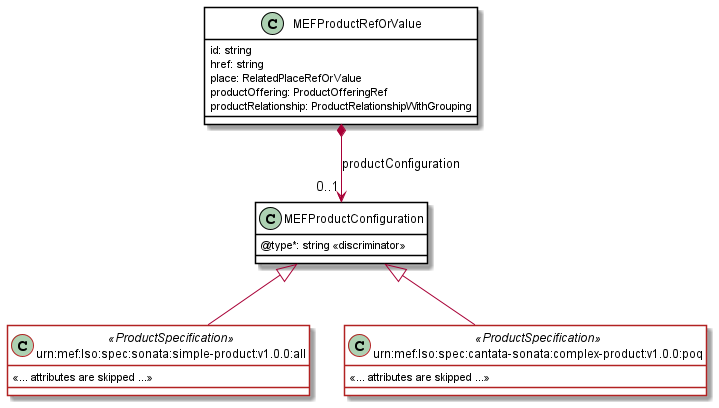


Figure The extension pattern

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 this (for POQ):

{

  "instantSyncQualification": true,

  "externalId": "BuyerPoq-00001",

  "provideAlternative": false,

  "projectId": "BuyerProjectX",

  "productOfferingQualificationItem": [

    {

POQ API part

      "id": "item-001",

      "action": "add",

      "product": {

        "productOffering": {

          "id": "000073"

        },

        "productConfiguration": {

          "@type": "urn:mef:lso:spec:sonata:simple-product:v1.0.0:all",

          "enumAttribute": "EXAMPLE\_VALUE\_ONE",

          "commonReusedInformationRate”: {

            "irValue" : "100",

            "irUnits" : "MBPS"

          }

        }

Simple Product part

      }

    }

  ]

}

## Format

Payload specifications MUST be delivered as a set of JSON schemas based on JSON schema draft 7 and encoded in YAML.

Payload specifications MUST contain the ”$id” property so that it can be uniquely identified. The syntax of this field is explained in section 4.5.

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

## 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 scheme no place explicitly defines the name of the type. The name of the type while blending (section 4.11) is decoded from the URN (see 4.5) which contains the name of the product in its 6th part. The file name with the schema SHOULD reflect the same product name.

## 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:

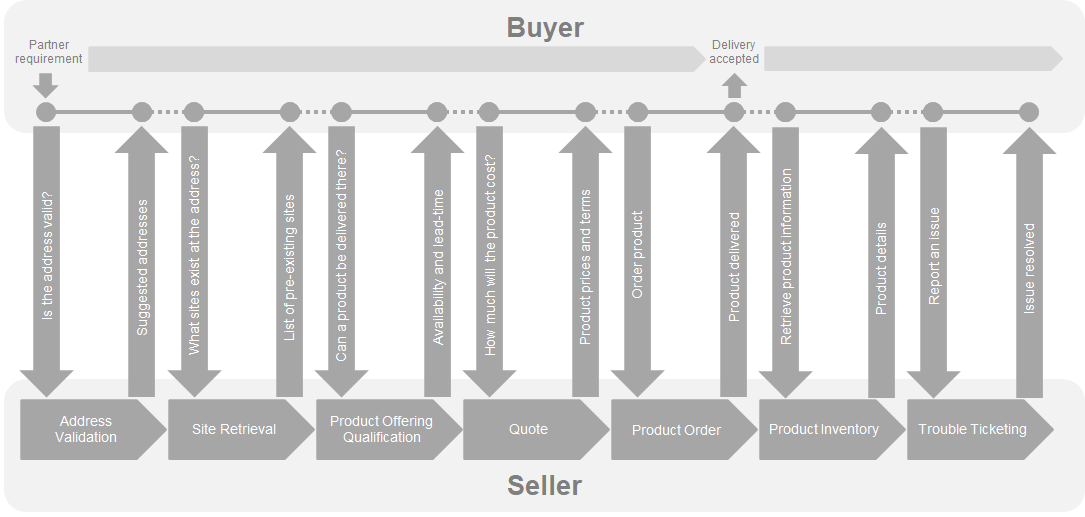


Figure LSO Cantata and LSO Sonata End-to-End Function Flow

* 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 in-stances 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.

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:

The model of the step must be a subset of the model of the next step. No attributes can be changed or deleted, they can only be potentially added. It is good practice, however, to keep the model consistent throughout all of the steps.

The next step can be more restrictive than the previous one in terms of the number of required attributes. Commonly, an Order requires more details to be provided compared to the POQ or Quote steps.

## “$id” / URN structure

The main purpose of this attribute is to uniquely identify a payload specification that must be used to interpret it by the other side of the API. It should contain information about the payload name, version, and use context (poq, quote, order…).

To ensure this uniqueness, MEF uses its registered URN space and specifies several requirements on its structure. The details are described on: [MEF Assigned Names and Numbers](https://wiki.mef.net/display/MANN/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 could be "epl:", "subscriber-ethernet-uni:", "evp-lan:", "internet-access:", etc.  This field uses the 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 could be "quote", "order", "inventory". If a single schema is used for all functions, the API Function should be "all". |

Table MEF LSO URN format

The urn of the Simple Product example is:

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

It is intended to be used in the context of the LSO Sonata IRP, the product name is “simple-product”, its version is 1.0.0 and this one flavor will serve all LSO Sonata functions.

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

This is the Complex Product, which can be used in the context of LSO Cantata and LSO Sonata, it’s version is 1.0.0 and this particular file will serve only in the context of the POQ function. The remaining functions must use other flavors which must be provided in the product specification.

## Common Model Reuse

A MEF Product Model project, while developed from MEF specifications, extracts any repetitive parts into common, reusable schemas that are referenced by the resulting payload specifications. Depending on the scope of reusability those can be general, technology, or product commons. They must be reused if a payload specification requires the usage of a matching property.

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 such reuse 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.

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 models schemas can be found in several places. There is a dedicated [MEF Wiki page](https://wiki.mef.net/display/CESG/Common+Data+Model) that provides updated information on respective artifacts or repositories.

Common schemas are delivered in the directory structure presented in Figure 10:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Figure Common model schemas directory structure

The structure is split initially into product and service (adding also the place to put schemas common to both). The next level is the technology division (e.g. carrierEthernet, ipIpvpn, sdWan, etc.) Each of the levels can have its common schemas defined.

## 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 ([6]):

*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 parts. However, it is recommended to have the additional requirements described in the accompanying documentation.

## Payload-Related Envelope Requirements

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

### 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), an attribute “relationshipType” is used. If the payload specification defines any relation to other products (or services) it must specify the required values for this attribute. An example of such information, taken from Access E-Line product specification (MEF W106, chapter 11, [6]) is presented inTable 2. Please note that additionally, this table specifies:

* Install/Change – if the relationship is mandatory or not to be provided for the 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 Product Relationship Types

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Product Relationship Role** | **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 set relationships is allowed. Such a use case can potentially mean moving the connection from one endpoint to another. This usually is forbidden and Delete and Create operations are needed 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 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).

### 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 2 APIs dedicated to use cases of Address Validation (MEF W121, [7]) or Site Retrieval (MEF W122, [8]) that serve the purpose of acquiring proper ids from the Seller. Additionally, the “place” has a mandatory subAttribute “role” which has the same function as the abovementioned “relationshipType” and the required 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.

Table Place Relationship Role

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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).

## Packaging

The payload package must be delivered as a zip of all relevant schemas in the proper directory structure (as described in the previous section). It should only contain common schemas that are referenced by the actual payload schemas. Figures below present example delivery packages of the example products.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Figure 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/carrrierEthernet/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.

Obraz zawierający stół

Opis wygenerowany automatycznie

Figure 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: "../../complexProductCommons/complexProductCommons.yaml#/definitions/ComplexProductCommon"

  - 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 ([6]). 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 commons.

## Static and dynamic binding

As mentioned in section 4.1 there are 2 building blocks of MEF LSO APIs: the functional product or service agnostic APIs (envelope), and the product or service-specific information model. They are delivered separately and must be bound together to form a fully functional API. This can be achieved in two ways: **Dynamic binding or Static binding**.

In the **Dynamic Binding** approach, the Seller exposes the API endpoint according to the envelope specification only. Looking at the exposed endpoint definition, 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 API (currently under development in MEF). The payload-specific part of the request is validated in runtime against a schema pointed to in the request. This approach allows for 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](https://github.com/MEF-GIT/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 exposed endpoint definition the Buyer can easily tell which products/services are supported. In such a case, 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 blending of example product specifications with POQ API would look like presented on 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 blended 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)

        # ...

## Blending Tool

LSO SDK releases provide a non-normative artifact of all product-oriented APIs pre-blended with all compatible product 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 blend them selectively with the relevant LSO APIs.

A manual blending of envelopes and payloads can be very time-consuming and error-prone. MEF is using a tool called “LSO Sonata Blending Tool” contributed by MEF member Amartus and made available as an open-source project in GitHub: [SonataBlendingTool](https://github.com/Amartus/SonataBlendingTool)**. It is a Java-based command-line tool. It allows both bulk and selective blending.**

With a single command, a developer can merge several payload specifications into one functional API schema. For example, a command that would blend 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 pre-blended pack of APIs with example payloads (as would be included in an LSO SDK) can be found in Appendix A.

The LSO Marketplace is planned to provide the blending functionality in a user-friendly way. The user will be able to select the Release, API, and desired Payload specifications to have them blended.

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

* basic schema syntax validation
* directory structure
* references validity
* URN syntax (starting with: “urn:mef”, payload name type taken from the sixth position)

# Non-MEF LSO Payload Documentation Requirements

Documentation is a requirement for MEF-Endorsed LSO Payloads to be included in the [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net). This section explains the documentation requirements and development guidelines.

## Document Format

The document must be provided using the MEF HTML template on the MEF Wiki provided for this purpose. It is recommended that documentation also be included within the documented schema in the descriptions.

## 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:

### Product/Service Description

An explanation should be provided of the underlying product or service, where and how it is used, and by what type of entities. If any market information is available, that should be included. Also, if the product or service is based on a standard, information should be included regarding the standard and the SDO behind it.

### 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.

### Concepts and Information Model Objects

The documentation should provide as much information as possible on the underlying concepts and attributes of the product or service, and any Information Modelling – even if only at a very high level.

### Use Cases

As much detail around the use cases for the product or service should be provided including the typical interaction between the entities using and providing the product/service. For example, if the product is designed to be purchased by an enterprise in an on-demand model or if the service needs to stream data based on policies etc.

### Internal Product/Service Dependencies

If there are any internal dependencies between objects and attributes, as described in section 4.7, they should also be described in the documentation.

## Payload-Related Envelope Requirements

Section 4.8 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.

## Buyer-Seller Onboarding Information.

The aim of MEF LSO APIs is 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 which cannot be understood directly from the Non-MEF Payload schema. The process of this bilateral agreement is often referred to as onboarding.

## 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 [6] in chapter 8.

Table 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 |

## 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

## 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 ([6]) 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](http://www.postman.com) collections.

# References

1. MEF 26.2, External Network Network Interfaces (ENNI) and Operator Service Attributes, August 2016
2. MEF 51.1, Operator Ethernet Service Definitions, December 2018
3. MEF 55.1, Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, January 2021
4. MEF 79, Address, Service Site, and Product Offering Qualification Management Requirements and Use Cases, November 2019
5. MEF W101, LSO Legato Service Provisioning Specification-Carrier Ethernet Schema Guide TODO
6. MEF W106, LSO Sonata Product Specification - Access E-Line - Schema Guide, June 2021
7. MEF W121, LSO Cantata and LSO Sonata Address Management API - Developer Guide, January 2022
8. MEF W122, LSO Cantata and LSO Sonata Site Management API - Developer Guide, January 2022
9. [RFC 2141](https://www.ietf.org/rfc/rfc2141.txt), URN Syntax, May 1997
10. [RFC 2396](https://www.ietf.org/rfc/rfc2396.txt), Uniform Resource Identifiers (URI): Generic Syntax, August 1998
11. LSO Payload Examples

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

* Simple Product:



* Complex Product



* Static binding:



1. Proposal Form for Non-MEF LSO Product Payload

If a company is interested in having its product or service schema posted in the [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net) as a non-MEF LSO Payload (MEF-Endorsed or Partner-Specific), it can submit a proposal to MEF via the LSO Developer Community Manager. 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 should include:

* The name of the company 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 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 blended
* Preferred type of MEF Payload (MEF-Endorsed or Partner-Specific)
* A short business description of the product or service
* An initial draft in the form of JSON schema or UML diagram (if available)

The application can be submitted to the LSO Developer Community Manager:   
[community\_manager@mef.net](mailto:community_manager@mef.net) only by a MEF member company.

1. 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 [Wiki page](https://wiki.mef.net/display/CESG/MEF+LSO+Developer+Community). Taking [Sonata SDK Billie release (public)](https://github.com/MEF-GIT/MEF-LSO-Sonata-SDK/tree/billie-final), for example, product schemas can be found in the \productSchema directory. Supporting documentation can be found in the [extended version of the SDK](https://github.com/MEF-GIT/MEF-LSO-Sonata-SDK-extended/tree/billie-final) release, available only MEF-members, 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.
* [LSO Marketplace](file:///C:\Users\Michał\Desktop\MEF\_Handbook\lso.mef.net)

This type is provided as a MEF published standard which is delivered by MEF community project(s). This process may be long-lasting yet brings broad MEF knowledge and industry consensus. The process involves many steps and deliverables. It usually takes:

* 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 Model
  + service’s information 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 Model
  + product’s information model provided in a form of a JSON schema file and supporting documentation
  + based on the service information 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) or separate documents (e.g. MEF 26.2 and MEF 51.1 for Carrier Ethernet). Service Model and Product Model specifications come as separate standards. Figure 13 presents the standardization steps together with examples of standards that delivered them.

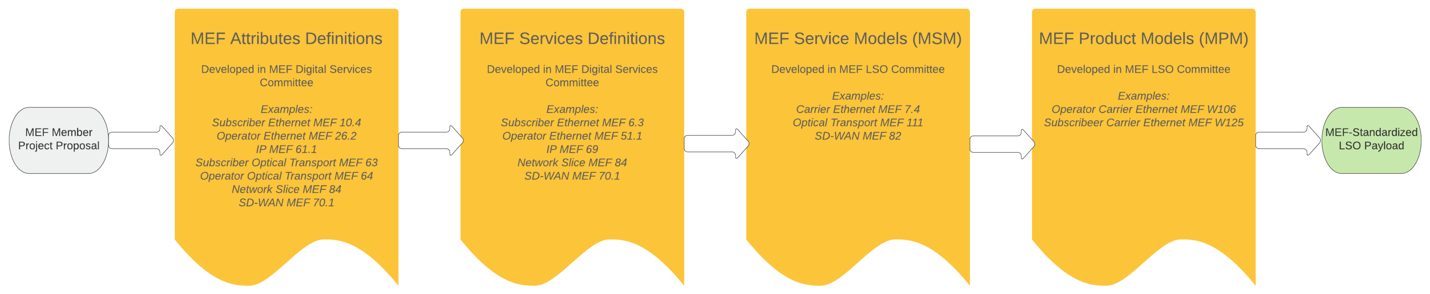


Figure MEF-Standard LSO Payload development process

* 1. MEF-Standardized Payloads Roadmap

The roadmaps for the next releases per IRP can be found on the LSO Marketplace. Related product or service specification deliverables are mentioned there.

The plans and current progress of respective projects can also be found via the LSO Marketplace.