Working Draft

MEF W124 v0.2

LSO Cantata and LSO Sonata Trouble Ticket Management API - Developer Guide

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

January 2022

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 member will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor
- (c) any form of relationship between any MEF member and the recipient or user of this document.

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

Copyright

Table of Contents

List of Contributing Members

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

Member

Table 1. Contributing Members

1. Abstract

This standard is intended to assist implementation of the Trouble Ticketing functionality defined for the LSO Cantata and LSO Sonata Interface Reference Points (IRPs), for which requirements and use cases are defined in MEF 113 *Trouble Ticketing Requirements and Use Cases* [MEF113]. This standard consists of this document and complementary API definitions for Trouble Ticket Management and Trouble Ticket Notification.

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

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

- `productApi/troubleTicket/troubleTicketManagement.api.yaml`
- `productApi/troubleTicket/troubleTicketNotification.api.yaml`

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

- `productApi/troubleTicket/troubleTicketManagement.api.yaml`
- `productApi/troubleTicket/troubleTicketNotification.api.yaml`

The Trouble Ticket API is defined using OpenAPI 3.0 [OAS-V3]

2. Terminology and Abbreviations

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

Term	Description	Reference
Application	In the context of LSO, API describes one of the Management	[MEF55.1]

Program Interface (API)	Interface Reference Points based on the requirements specified in an Interface Profile, along with a data model, the protocol that defines operations on the data and the encoding format used to encode data according to the data model. In this document, API is used synonymously with REST API	
Buyer	In the context of this document, denotes the organization or individual acting as the customer in a transaction over a Cantata (Customer <-> Service Provider) or Sonata (Service Provider <-> Partner) Interface	This document; adapted from [MEF80]
Incident	An entry within a Seller's tracking system created by the context of this document, denotes a situation that is not part of normal operationSeller, which contains information about a Situation in the Seller's network that has a possible negative impact on the operability of the network ona Product for one or more Buyers	[MEF113]
Issue	In the context of this document, denotes a problem with a Product as experienced by the Buyer that is not part of normal operation.	[MEF113]
Notification	A message sent from the Seller to the Buyer to inform about an event that has occurred in regard to a specific instance of a Ticket or an Incident	[MEF113]
Requesting Entity	The business organization that is acting on behalf of one or more Buyers. In the most common case, the Requesting Entity represents only one Buyer and these terms are then synonymous	[MEF79]
Responding Entity	The business organization that is acting on behalf of one or more Sellers. In the most common case, the Responding Entity represents only one Seller and these terms are then synonymous	[MEF79]
REST API	Representational State Transfer. REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems.	[REST]
Seller	In the context of this document, denotes the organization acting as the supplier in a transaction over a Cantata (Customer <-> Service Provider) or Sonata (Service Provider <-> Partner) Interface	This document; adapted

		from [MEF80]
Situation	In the context of this document, denotes a problem that is not part of normal operation in the Seller's network that has a possible negative impact on the operability of a Product for one or more Buyers	[MEF113]
Ticket	An entry within a Seller's tracking system created by the Buyer (or a third party on behalf of the Buyer), which contains information about an Issue impacting normal operation of a Product, along with support interventions made by technical support staff, or third parties	[MEF113]
Trouble Ticketing	In the context of this document, denotes the management of both Tickets and Incidents	[MEF113]
Workorder	In the context of this document, denotes a set of tasks to be scheduled and performed under the responsibility of a Technician at a given location	[MEF113]

3. Compliance Levels

The key words **"MUST"**, **"MUST NOT"**, **"REQUIRED"**, **"SHALL"**, **"SHALL NOT"**, **"SHOULD"**, **"SHOULD NOT"**, **"RECOMMENDED"**, **"NOT RECOMMENDED"**, **"MAY"**, and **"OPTIONAL"** in this document are to be interpreted as described in BCP 14 (RFC 2119 [RFC2119], RFC 8174 [RFC8174]) when, and only when, they appear in all capitals, as shown here. All key words must be in bold text.

Items that are **REQUIRED** (contain the words **MUST** or **MUST NOT**) are labeled as **[Rx]** for required. Items that are **RECOMMENDED** (contain the words **SHOULD** or **SHOULD NOT**) are labeled as **[Dx]** for desirable. Items that are **OPTIONAL** (contain the words MAY or OPTIONAL) are labeled as **[Ox]** for optional.

4. Introduction

The Trouble Ticket API allows the Buyer to create, retrieve, and update Trouble Tickets as well as receive notifications and Trouble Tickets' updates. This allows managing issues and situations that are not part of normal operations of the Product provided by the Seller.

This standard specification document describes the Application Programming Interface (API) for Trouble Ticketing functionality of the LSO Cantata Interface Reference Point (IRP) and LSO Sonata IRP as defined in the *MEF 55.1 Lifecycle Service Orchestration*

(LSO): Reference Architecture and Framework [MEF55.1]. The LSO Reference Architecture is shown in Figure 1 with both IRPs highlighted.

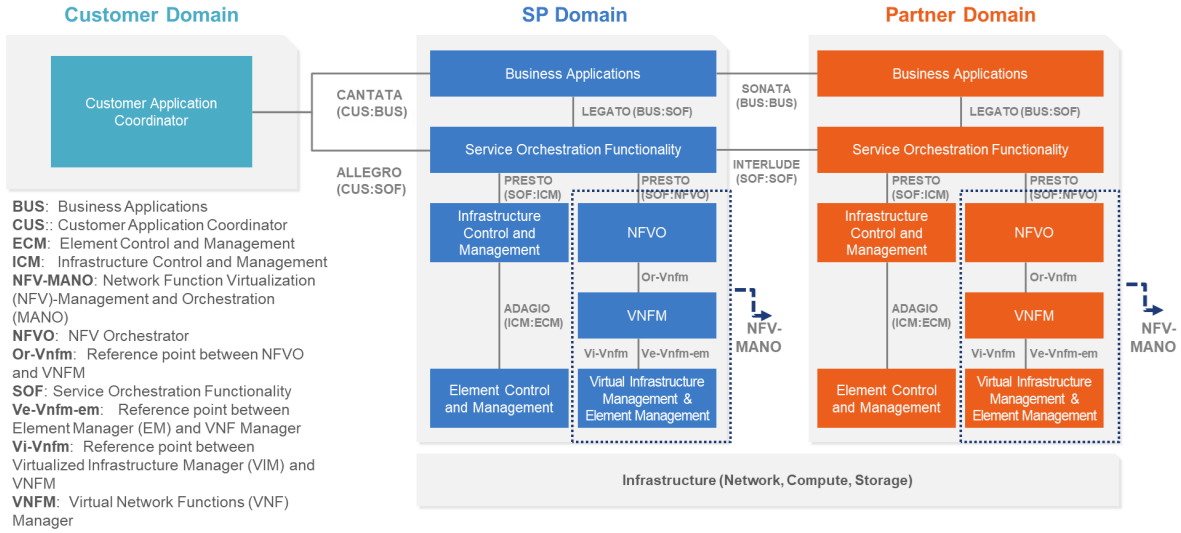


Figure 1. The LSO Reference Architecture

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

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

The business requirements and use cases for Trouble Ticketing are defined in MEF W113 *Trouble Ticketing Requirements and Use Cases* [MEF113]. MEF W113 defines use cases that cover Trouble Ticket, Incident, Appointment and WorkOrder. This API and Developer Guide covers the Trouble Ticket related use cases, basing on the [TMF621] Trouble Ticket API, Incident and WorkOrder. The support for Appointment use cases will be provided in future releases.

This document is structured as follows:

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

4.1. Conventions in the Document

- Code samples are formatted using code blocks. When notation `<< some text >>` is used in the payload sample it indicates that a comment is provided instead of an example value and it might not comply with the OpenAPI definition.
- Model definitions are formatted as in-line code (e.g. `TroubleTicket`).
- In UML diagrams the default cardinality of associations is `0..1`. Other cardinality markers are compliant with the UML standard.
- In the API details tables and UML diagrams required attributes are marked with a `*` next to their names.
- In UML sequence diagrams `{{variable}}` notation is used to indicates a variable to be substituted with a correct value.

4.2. Relation to Other Documents

This API implements the Trouble Ticket related requirements and use cases that are defined in MEF 113 [MEF113]. The API definition builds on *TMF621 Trouble Ticket API REST Specification R19.0.1* [TMF621].

4.3. Approach

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

- Generic API framework
- Product-independent information (Function-specific information and Function-specific operations)
- Product-specific information (MEF product specification data model)

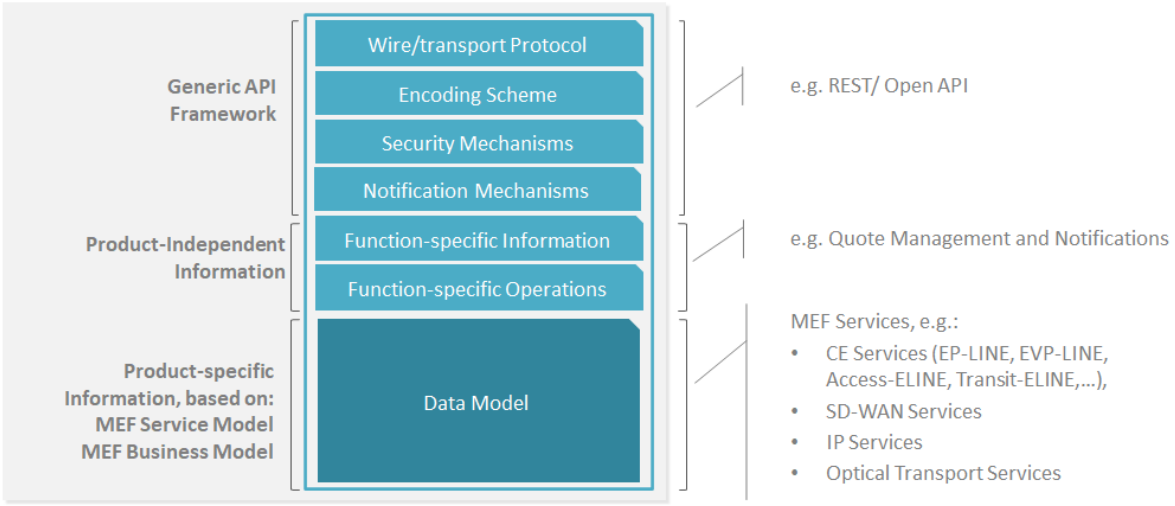


Figure 2. Cantata and Sonata API framework

The essential concept behind the framework is to decouple the common structure, information and operations from the specific product information content.
Firstly, the Generic API Framework defines a set of design rules and patterns that are applied across all Cantata or Sonata APIs.
Secondly, the product-independent information of the framework focuses on a model of a particular Cantata or Sonata functionality and is agnostic to any of the product specifications.
Finally, the product-specific information part of the framework focuses on MEF product specifications that define business-relevant attributes and requirements for trading MEF subscriber and MEF operator services.

The Trouble Ticket is product-agnostic in it's nature and is not intended to carry any product-specific payloads. It only references product from the inventory by `id`. It operates using the Generic API Framework and the Function-specific Information and Operations.

4.5. High-Level Flow

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

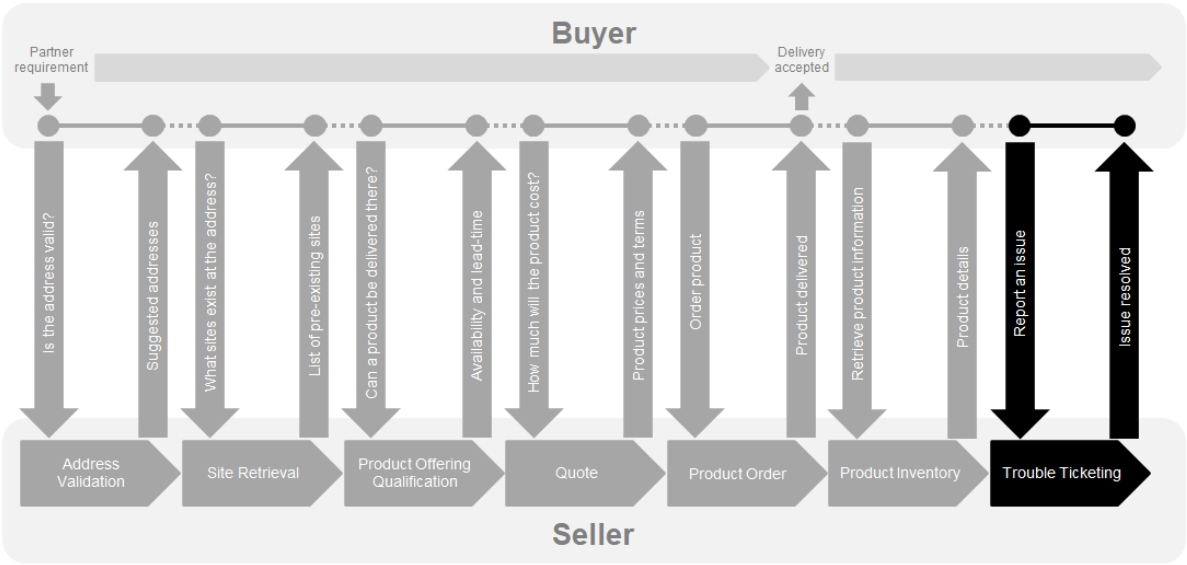


Figure 3. Cantata and Sonata End-to-End Function Flow

- Address Validation:
  - Allows the Buyer to retrieve address information from the Seller, including exact formats, for addresses known to the Seller.
- Site Retrieval:
  - Allows the Buyer to retrieve Geographic Site information including exact formats for Geographic Sites known to the Seller.
- Product Offering Qualification (POQ):

    - Allows the Buyer to check whether the Seller can deliver a product or set of products from among their product offerings at the geographic address or a Geographic Site specified by the Buyer; or modify a previously purchased product.
- Quote:
    - Allows the Buyer to submit a request to find out how much the installation of an instance of a Product Offering, an update to an existing Product, or a disconnect of an existing Product will cost.
- Product Order:
    - Allows the Buyer to request the Seller to initiate and complete the fulfillment process of an installation of a Product Offering, an update to an existing Product, or a disconnect of an existing Product at the address defined by the Buyer.
- Product Inventory:
    - Allows the Buyer to retrieve the information about existing Product instances from Seller's Product Inventory.
- Trouble Ticketing:
    - Allows the Buyer to create, retrieve, and update Trouble Tickets as well as receive notifications about Incidents' and Trouble Tickets' updates. This allows managing issues and situations that are not part of normal operations of the Product provided by the Seller.

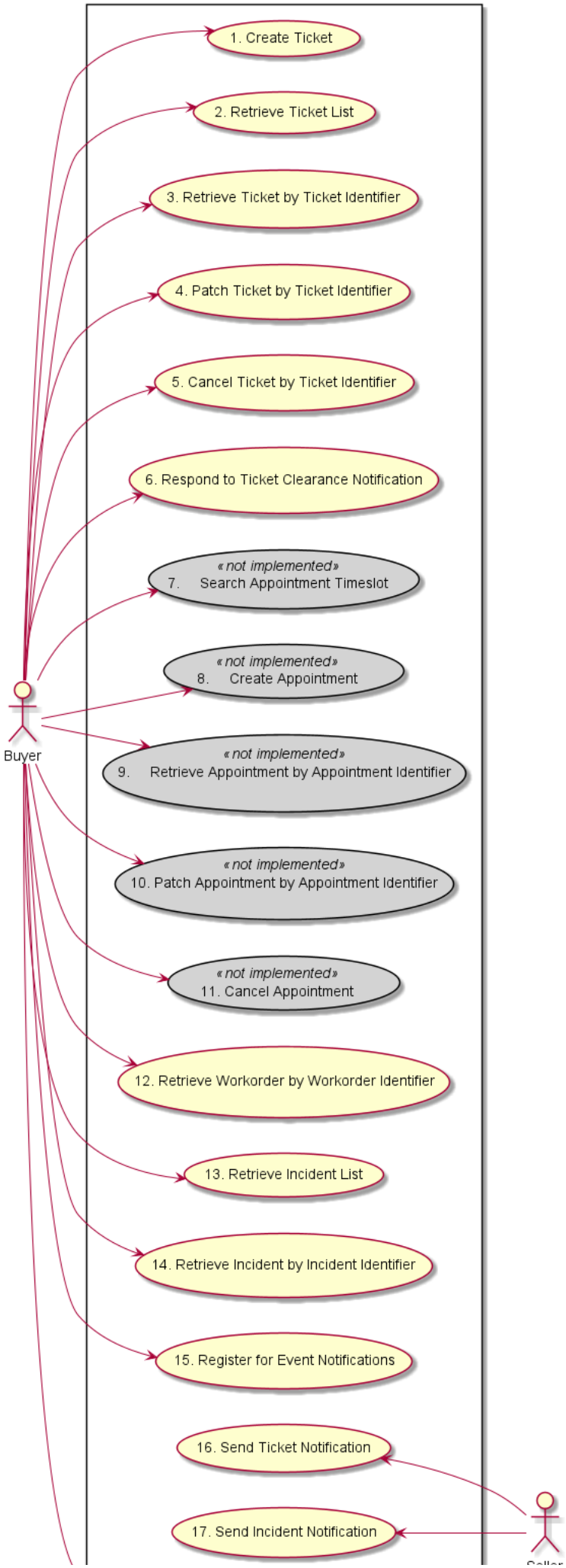
5. API Description

This section presents the API structure and design patterns. It starts with the high-level use cases diagram. Then it describes the REST endpoints with use case mapping. Next, it gives an overview of the API resource model.

5.1. High-level use cases

Figure 4 presents a high-level use case diagram as specified in MEF 113 [MEF113] in section 7. This picture aims to help understand the endpoint mapping. Use cases are described extensively in chapter 6.

Note: As stated earlier this API does not implement the Appointment related use cases. The diagram below lists all use cases defined in MEF 113 to highlight which of them are implemented. For easier requirements matching this documents keep the original numbering. The remaining use cases will be delivered by separate APIs delivered by MEF in future releases.


1. Create Ticket
2. Retrieve Ticket List
3. Retrieve Ticket by Ticket Identifier
4. Patch Ticket by Ticket Identifier
5. Cancel Ticket by Ticket Identifier
6. Respond to Ticket Clearance Notification
«not implemented»
7. Search Appointment Timeslot
«not implemented»
8. Create Appointment
«not implemented»
9. Retrieve Appointment by Appointment Identifier
«not implemented»
10. Patch Appointment by Appointment Identifier
«not implemented»
11. Cancel Appointment
12. Retrieve Workorder by Workorder Identifier
13. Retrieve Incident List
14. Retrieve Incident by Incident Identifier
15. Register for Event Notifications
16. Send Ticket Notification
17. Send Incident Notification
Buyer

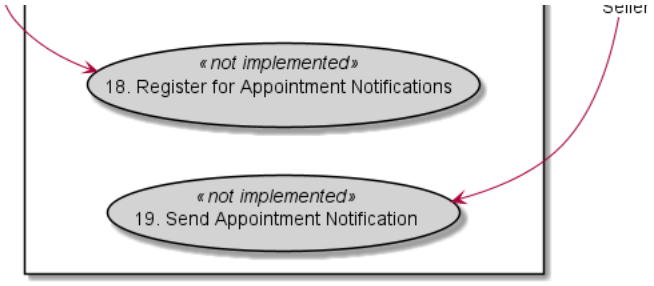


Figure 4: Use cases

5.2. API Endpoint and Operation Description

5.2.1. Seller side API Endpoints

Base URL for Cantata: `https://{{server}}:{{port}}{{?/seller_prefix}}/mefApi/cantata/troubleTicket/v2/`

Base URL for Sonata: `https://{{server}}:{{port}}{{?/seller_prefix}}/mefApi/sonata/troubleTicket/v2/`

The following API endpoints are implemented by the Seller and allow the Buyer to create, retrieve, modify Trouble Tickets and register for Notifications. The endpoints and corresponding data model are defined in `productApi/troubleTicket/troubleTicketManagement.api.yaml`.

API endpoint	Description	MEF 113 Use Case mapping
`POST /troubleTicket`	A request initiated by the Buyer to create a Ticket in the Seller's system to report an Issue experienced by the Buyer or their end user.	UC 1: Create Ticket
`GET /troubleTicket`	The Buyer requests a list of Tickets from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Tickets.	UC 2: Retrieve Ticket List
`GET /troubleTicket/{{id}}`	The Buyer requests detailed information about a single Ticket based on a Ticket Identifier.	UC 3: Retrieve Ticket by Ticket Identifier
`PATCH /troubleTicket/{{id}}`	A request by the Buyer to patch/partial up-date a Ticket created by the Buyer in the Seller's system.	UC 4: Patch Ticket by Ticket Identifier

API endpoint	Description	MEF 113 Use Case mapping
`POST /troubleTicket/{{id}}/cancel`	A request by the Buyer to cancel a Ticket created by the Buyer in the Seller's system.	UC 5: Cancel Ticket by Ticket Identifier
`POST /troubleTicket/{{id}}/close`	A request from the Buyer confirming whether they agree that a Ticket created by the Buyer in the Seller's system can be closed, since the reported Issue is no longer observed. This request is the action taken by a Buyer after receiving a Ticket Notification from the Seller with Ticket Notification Event Type `TroubleTicketResolvedEvent`.	UC 6: Respond to Ticket Clearance Notification
`POST /troubleTicket/{{id}}/reopen`	A request from the Buyer rejecting that a Ticket created by the Buyer in the Seller's system can be closed, because the reported Issue is still observed. This request is the action taken by a Buyer after receiving a Ticket Notification from the Seller with Ticket Notification Event Type `TroubleTicketResolvedEvent`.	UC 6: Respond to Ticket Clearance Notification
`GET /workOrder/{{id}}`	The Buyer requests detailed information about a Workorder based on a Workorder Identifier.	UC 12. Retrieve Workorder by Workorder Identifier
`POST /hub`	The Buyer requests to subscribe to notifications.	UC 15: Register for Event Notifications
`GET /hub/{{id}}`	A request initiated by the Buyer to retrieve the details of the notification subscription.	UC 15: Register for Event Notifications
`DELETE /hub/{{id}}`	A request initiated by the Buyer to instruct the Seller to stop sending notifications.	UC 15: Register for Event Notifications

Table 2. Seller side mandatory API endpoints

[R1] The Seller **MUST** support API endpoints listed in Table 2. [MEF113 R1]

API endpoint	Description	MEF 113 Use Case mapping
`GET /incident`	The Buyer requests a list of Incidents from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Incidents.	UC 13. Retrieve Incident List
`GET /incident/{{id}}`	The Buyer requests detailed information about a single Incident based on an Incident Identifier.	UC 14. Retrieve Incident by Incident Identifier

Table 3. Seller side optional API endpoints

[O1] The Seller **MAY** support API endpoints listed in Table 3. [MEF113 O1]

5.2.2. Buyer side API Endpoints

Base URL for Cantata: `https://{{server}}:{{port}}{{?/buyer_prefix}}/mefApi/cantata/troubleTicketNotification/v2/`

Base URL for Sonata: `https://{{server}}:{{port}}{{?/buyer_prefix}}/mefApi/sonata/troubleTicketNotification/v2/`

The following API Endpoints are used by the Seller to post notifications to registered listeners. The endpoints and corresponding data model are defined in `productApi/troubleTicket/troubleTicketNotification.api.yaml`

API Endpoint	Description	MEF 113 Use Case Mapping
`POST /listener/troubleTicketAttributeValueChangeEvent`	A request initiated by the Seller to notify the Buyer on `TroubleTicket` attribute value change.	UC 14: Send Ticket Notification
`POST /listener/troubleTicketStatusChangeEvent`	A request initiated by the Seller to notify the Buyer on `TroubleTicket` state change.	UC 14: Send Ticket Notification
`POST /listener/troubleTicketResolvedEvent`	A request initiated by the Seller to notify the Buyer on `TroubleTicket` reaching the `resolved` state.	UC 14: Send Ticket Notification

API Endpoint	Description	MEF 113 Use Case Mapping
`POST /listener/troubleTicketInformationRequiredEvent`	A request initiated by the Seller to notify the Buyer that and additional information is required for further Ticket processing	UC 14: Send Ticket Notification

Table 4. Buyer side mandatory API endpoints

[R2] The Seller **MUST** support API endpoints listed in Table 4. [MEF113 R1]

API Endpoint	Description	MEF 113 Use Case Mapping
`POST /listener/incidentCreatedEvent`	A request initiated by the Seller to notify the Buyer on `Incident` creation	17. Send Incident Notification
`POST /listener/incidentAttributeValueChangeEvent`	A request initiated by the Seller to notify the Buyer on `Incident` attribute value change.	17. Send Incident Notification
`POST /listener/incidentClosedEvent`	A request initiated by the Seller to notify the Buyer on `Incident` reaching the `closed` state.	17. Send Incident Notification
`POST /listener/incidentStatusChangeEvent`	A request initiated by the Seller to notify the Buyer on `Incident` state change.	17. Send Incident Notification

Table 5. Buyer side optional API endpoints

[O2] The Seller **MAY** support API endpoints listed in Table 5. [MEF113 O1]

[CR1]<([O1], [O2]) If any of endpoints implementing Use Cases 13, 14 or 17 is supported, then all endpoints implementing Use Cases 13, 14 and 17 **MUST** be supported. [MEF113 [CR1]<[O1]]

5.3. Specifying the Buyer ID and the Seller ID

A business entity willing to represent multiple Buyers or multiple Sellers must follow requirements of MEF 79 [MEF79] chapter 8.8, which states:

> For requests of all types, there is a business entity that is initiating an Operation (called a Requesting Entity) and a business entity that is responding to this request (called the Responding Entity). In the simplest case, the Requesting Entity is the Buyer and the Responding Entity is the Seller. However, in some cases, the Requesting Entity may represent more than one Buyer and similarly, the Responding Entity may represent more than one Seller.
>
> While it is outside the scope of this specification, it is assumed that the Requesting Entity and the Responding Entity are aware of each other and can authenticate requests initiated by the other party. It is further assumed that both the Buying Entity and the Requesting Entity know:
>
> a) the list of Buyers the Requesting Entity represents when interacting with this Responding Entity; and
> b) the list of Sellers that this Responding Entity represents to this Requesting Entity.

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

[R3] If the Requesting Entity has the authority to represent more than one Buyer the request **MUST** include `buyerId` query parameter that identifies the Buyer being represented [MEF79 R80]

[R4] If the Requesting Entity represents precisely one Buyer with the Responding Entity, the request **MUST NOT** specify the `buyerId` [MEF79 R81]

[R5] If the Responding Entity represents more than one Seller to this Buyer the request **MUST** include `sellerId` query parameter that identifies the Seller with whom this request is associated [MEF79 R82]

[R6] If the Responding Entity represents precisely one Seller to this Buyer, the request **MUST NOT** specify the `sellerId` [MEF79 R83]

[R7] If `buyerId` or `sellerId` attributes were specified in the request same attributes **MUST** be used in the notification payload.

5.4. Model Structural Validation

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

[R8] Implementations **MUST** use payloads that conform to these definitions.

5.5. Security Considerations

There must be an authentication mechanism whereby a Seller can be assured who a Buyer is and vice-versa. There must also be authorization mechanisms in place to control what a particular Buyer or Seller is allowed to do and what information may be obtained. However, the definition of the exact security mechanism and configuration is outside the scope of this document. It is being worked on by a separate MEF Project (MEF W128).

6. API Interactions and Flows

This section provides a detailed insight into the API functionality, use cases, and flows. It starts with Table 6 presenting a list and short description of all business use cases then presents the variants of end-to-end interaction flows, and in following subchapters describes the API usage flow and examples for each of the use cases.

Use Case #	Use Case Name	Use Case Description
1	Create Ticket	A request initiated by the Buyer to create a Ticket in the Seller's system to report an Issue experienced by the Buyer or their end user.
2	Retrieve Ticket List	The Buyer requests a list of Tickets from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Tickets.
3	Retrieve Ticket by Ticket Identifier	The Buyer requests detailed information about a single Ticket based on a Ticket Identifier.
4	Patch Ticket by Ticket Identifier	A request by the Buyer to patch/partial update a Ticket created by the Buyer in the Seller's system.
5	Cancel Ticket by Ticket Identifier	A request by the Buyer to cancel a Ticket created by the Buyer in the Seller's system.
6	Respond to Ticket Clearance Notification	A request from the Buyer confirming whether they agree that a Ticket created by the Buyer in the Seller's system can be closed, since the reported Issue is no longer observed. This request is the action taken by a Buyer after receiving a Ticket Notification from the Seller with Ticket Notification Event Type `TroubleTicketResolvedEvent`.

Use Case #	Use Case Name	Use Case Description
12	Retrieve Workorder by Workorder Identifier	The Buyer requests detailed information about a Workorder based on a Workorder Identifier.
13	Retrieve Incident List	The Buyer requests a list of Incidents from the Seller based on a set of specified filter criteria. The Seller returns a summarized list of Incidents.
14	Retrieve Incident by Incident Identifier)	The Buyer requests detailed information about a single Incident based on an Incident Identifier.
15	Register for Event Notifications	The Buyer requests to subscribe to Ticket Notifications and optionally Incident Notifications.
16	Send Ticket Notification	The Seller sends a notification regarding a Ticket to the Buyer (if registered).
17	Send Incident Notification	The Seller sends a notification regarding an Incident to the Buyer (if registered).

Table 6. Use cases description

The detailed business requirements of each of the use cases are described in sections 7 and 8 of MEF 113 [MEF113].

6.1. Use case 1: Create Ticket

This is the initial step for Trouble Ticket processing.

6.1.1. Interaction flow

The flow of this use case is very simple and is described in Figure 5.

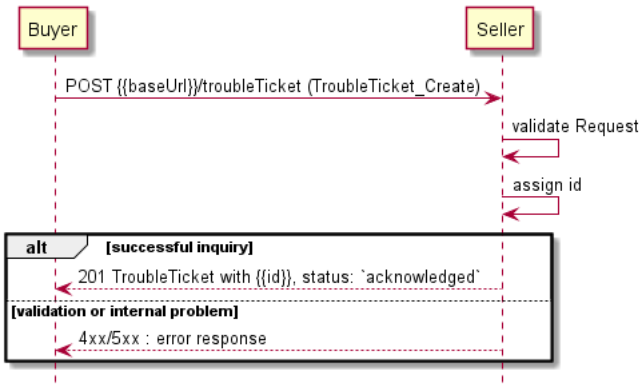


Figure 5: Use Case 1 - Trouble Ticket create request flow

The Buyer sends a request with a `TroubleTicket_Create` type in the body. The Seller performs request validation, assigns an `id`, and returns `TroubleTicket` type in the response body, with a `status` set to `acknowledged`. From this point, the Trouble Ticket is ready for further processing. The Buyer must track the progress of the process by subscribing for notifications (see chapter 6.10). The flow example with the use of Notifications is presented in Figure 6

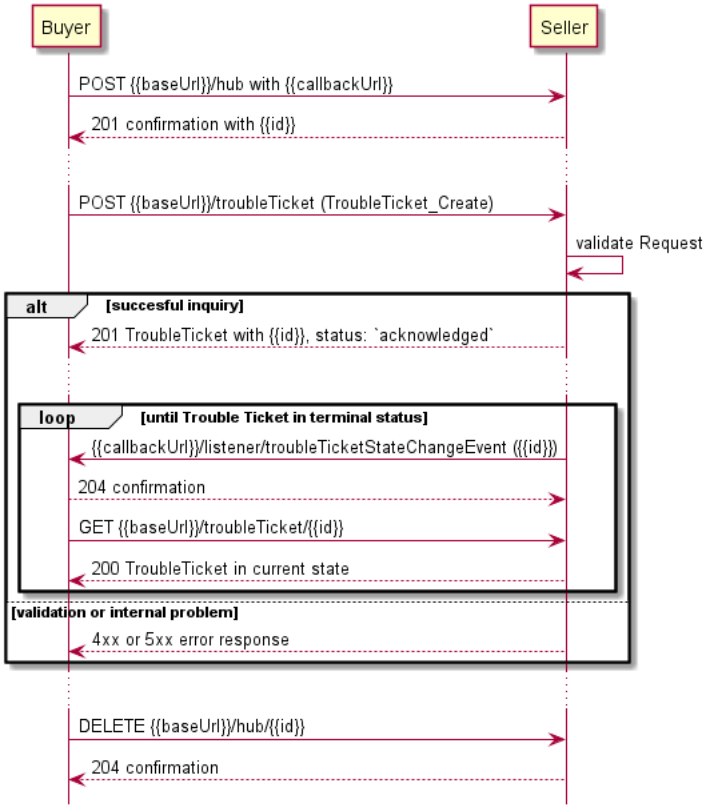


Figure 6: Trouble Ticket progress tracking - Notifications

Note: The context of notifications is not a part of the considered use case itself. It is presented to show the big picture of end-to-end flow. This applies also to all further use case flow diagrams with notifications.

6.1.1. Create Trouble Ticket - Request

Figure 7 presents the data model of the Trouble Ticket. The model of the request message (`TroubleTicket_Create`) is a subset of the `TroubleTicket` model and contains only attributes that can (or must) be set by the Buyer. The Seller then enriches the entity in the response with additional information. For visibility of these shared attributes, the `TroubleTicket_Common` has been introduced. Though, it is not to be used directly in the payload.

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

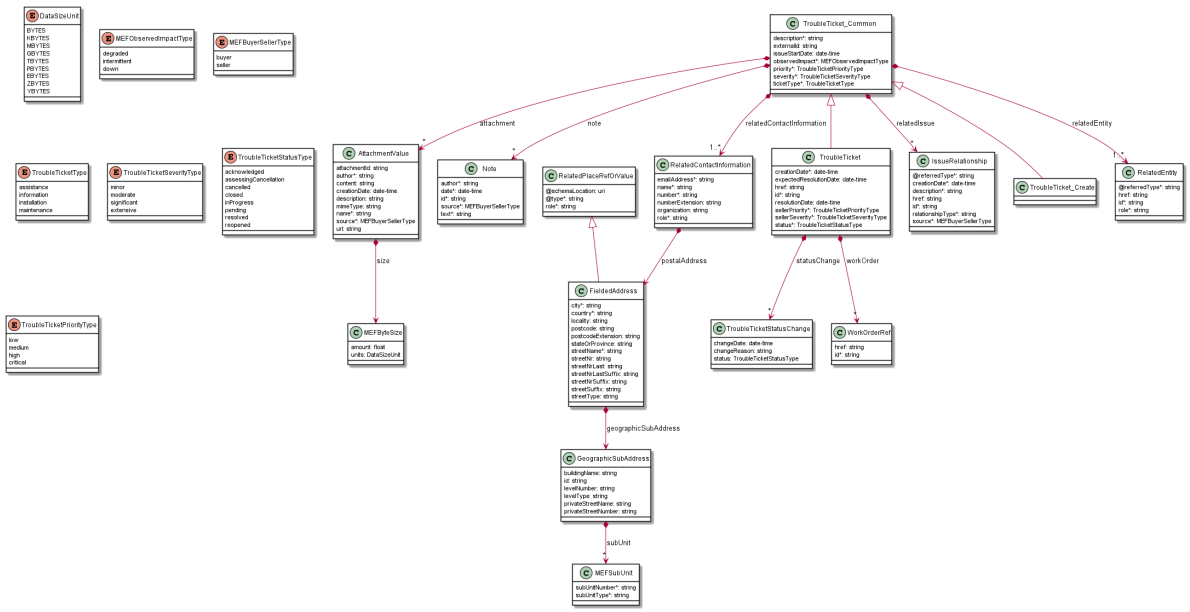


Figure 7: Create Trouble Ticket Model

The snippet below presents an example of the Create Trouble Ticket Request:

`TroubleTicket` Create


```
{
  "description": "Connection is lost",
  "externalId": "BuyerTicket-123",
  "issueStartDate": "2021-06-02T14:21:11.090Z",
  "priority": "critical",
  "severity": "extensive",
  "ticketType": "failure",
  "attachment": [
    {
      "attachmentId": "att-001",
      "author": "John Example",
      "creationDate": "2021-06-02T14:21:11.090Z",
      "description": "Print screen from the assurance system",
      "mimeType": "image/jpeg",
      "name": "Alarm",
      "url": "https://example.com/documents/00000000-0000-1111-2222-000000001111",
      "size": {
        "amount": 5.3,
        "units": "MBYTES"
      },
      "source": "buyer"
    }
  ],
  "note": [
    {
      "id": "note-1",
      "author": "John Example",
      "date": "2021-06-02T14:25:11.090Z",
      "source": "buyer",
```

```
        "text": "Couldn't reach the support on phone."
      }
    ],
    "relatedEntity": [ <<A relation to a Product that this Ticket refers to>>
      {
        "id": "01494079-6c79-4a25-83f7-48284196d44d",
        "role": "Issue Source",
        "@referredType": "Product"
      }
    ],
    "relatedContactInformation": [
      {
        "emailAddress": "john.example@example.com",
        "name": "John Example",
        "number": "+12-345-678-90",
        "organization": "Buyer Example Co.",
        "role": "reporterContact"
      }
    ]
  }
```

[R9] The Buyer's Create request **MUST** include the following attributes: [MEF113 R19]

- `description`
- `observedImpact`
- `priority`
- `relatedContactInformation` item with a `role` set to `reporterContact`
- `relatedEntity` - (pointer to related Product instance)
- `severity`
- `ticketType`

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

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

Note: The `relatedEntity` attribute is used to provide the related product `id`. It is done by setting the additional `@referredType` to `Product`. This follows the TMF pattern which enables compliance and allows referring also other potential types in MEF (e.g. `Service`). In this version, the only type that is mentioned in the implemented requirements document is the `Product` and to ease the request `RelatedEntity.@ReferredType` and the `relatedEntityType` in the filter criteria have a default value: `Product`.

[R10] If the `attachment` is provided, either the `attachment.url` or (`attachment.content` and `attachment.mimeType`) **MUST** be specified. [MEF113 R8], [MEF113 R9], [MEF113 R20]

6.1.2. Create Trouble Ticket - Response

The Seller responds with a `TroubleTicket` type, which adds some attributes to the `TroubleTicket_Create` that was used in the Buyer's request.

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

The following snippet presents the Seller's response. It has the same structure as in the retrieve by identifier operation.


```
{
  "id": "00000000-4444-5555-6666-000000000987",
  "href": "{{baseUrl}}/troubleTicket/00000000-4444-5555-6666-000000000987",
  "creationDate": "2021-06-02T20:56:08.559Z",
  "expectedResolutionDate": "2021-06-03T20:56:08.559Z",
  "lastUpdate": "2021-06-02T20:56:08.559Z",
  "sellerPriority": "critical",
  "sellerSeverity": "extensive",
  "status": "acknowledged",
  "description": "Connection is lost", << as provided by the Buyer >>
  "externalId": "BuyerTicket-123", << as provided by the Buyer >>
  "issueStartDate": "2021-06-02T14:21:11.090Z", << as provided by the Buyer >>
  "priority": "critical", << as provided by the Buyer >>
  "severity": "extensive", << as provided by the Buyer >>
  "ticketType": "failure", << as provided by the Buyer >>
  "attachment": [
    { << as provided by the Buyer >>
      "attachmentId": "att-001",
      "author": "John Example",
      "creationDate": "2021-06-02T14:21:11.090Z",
      "description": "Print screen from the assurance system",
      "mimeType": "image/jpeg",
      "name": "Alarm",
      "url": "https://example.com/documents/00000000-0000-1111-2222-000000001111",
      "size": {
        "amount": 5.3,
        "units": "MBYTES"
      },
      "source": "buyer"
    }
  ],
  "note": [
    {<< as provided by the Buyer >>
      "id": "note-1",
      "author": "John Example",
      "date": "2021-06-02T14:25:11.090Z",
      "source": "buyer",
      "text": "Couldn't reach the support on phone."
    }
  ],
  "relatedEntity": [
    {<< as provided by the Buyer >>
      "id": "01494079-6c79-4a25-83f7-48284196d44d",
      "role": "Issue Source",
      "@referredType": "Product"
    }
  ],
  "relatedContactInformation": [
    {<< as provided by the Buyer >>
      "emailAddress": "john.example@example.com",
      "name": "John Example",
      "number": "+12-345-678-90",
      "organization": "Buyer Example Co.",
      "role": "reporterContact"
    },
    {<< a new item appended by the Buyer >>
      "emailAddress": "Seller.TicketContact@example.com",
      "name": "Seller Ticket Contact",
      "number": "+98-765-432-10",
      "organization": "Seller Example Co.",
      "role": "sellerTicketContact"
    }
  ],
  "relatedIssue": [
    {
      "@referredType": "TroubleTicket",
      "id": "00000000-1234-4321-1111-00000000888",
      "creationDate": "2021-06-02T20:56:08.559Z",
      "description": "The issue is caused by.",
```

```
        "relationshipType": "caused by",
        "source": "seller"
      }
    ],
    "statusChange": [
      {
        "changeDate": "2021-06-02T20:56:08.560Z",
        "status": "acknowledged"
      }
    ]
  }
```

The response to the create request does not contain all possible attributes, for example, the `resolutionDate` is valid only in the future lifecycle of the Trouble Ticket.

[R11] The Seller's response **MUST** include all and unchanged attributes' values provided in the request. [MEF113 R23]

These attributes are indicated above with an appropriate comment: `<< as provided by the Buyer >>`.

[R12] The Seller **MUST** specify the following attributes in a response: [MEF113 R25]

- `creationDate`
- `id`
- `relatedContactInformation` - item with a `role` set to `sellerTicketContact`
- `relatedEntity` - (pointer to related Product instance)
- `sellerSeverity`
- `sellerPriority`
- `status`

[R13] The `status` of the Ticket in the Seller's response **MUST** be `acknowledged`. [MEF113 R24]

6.1.3. Trouble Ticket - Lifecycle

Figure 8 presents the Trouble Ticket state machine:

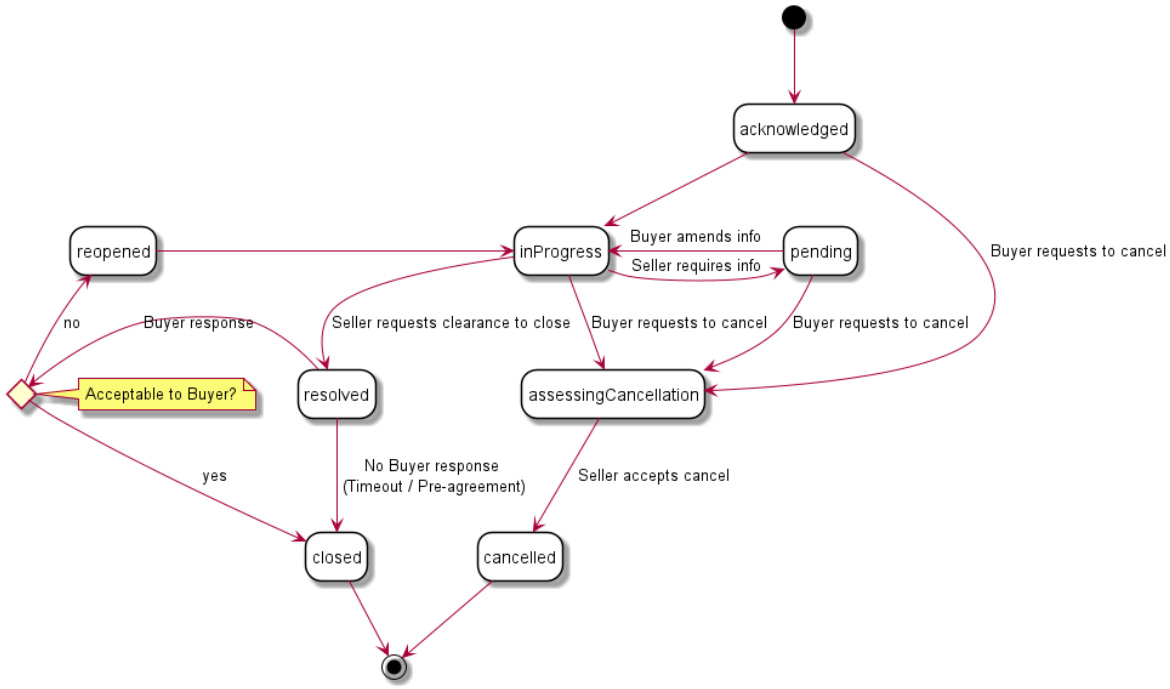


Figure 8: Trouble Ticket State Machine

After receiving the request, the Seller performs a validation of the message. If any problem is found an Error response is provided. If the validation passes a response is provided with `TroubleTicket` in `acknowledged` status. Then the Seller starts working on resolving the issue and moves the Trouble Ticket to `inProgress` state. From there, additional information might be required to proceed and the Trouble Ticket moves to `pending` until one is provided. The Trouble Ticket is set as `resolved` when the Seller claims the issue is fixed. From there the Buyer can either reopen or close the Ticket (use cases described in following sections). The Buyer may also request for a Trouble Ticket to be cancelled, while in `acknowledged`, `pending`, or `inProgress` state.

Table 7 presents the mapping between the API `status` names (aligned with TMF) and the MEF 113 naming, together with statuses' description.

status	MEF 113 name	Description
`acknowledged`	ACKNOWLEDGED	A request to create a Ticket was received and accepted by the Seller. The Ticket has been validated and created by the Seller and allocated a unique `id`.
`inProgress`	IN_PROGRESS	The Ticket is in the process of being handled and investigated for resolution by the Seller.

status	MEF 113 name	Description
`resolved`	RESOLVED	The Buyer's Issue described in the Ticket was resolved by the Seller. The Seller is now waiting for the Buyer to confirm that the Issue they reported is no longer observed.
`closed`	CLOSED	The Buyer that created the Ticket has confirmed that the Issue they reported is no longer observed, or the pre-defined time frame (agreed upon between Buyer and Seller) for confirming that the Issue has been resolved has passed without a response by the Buyer. This is a terminal state.
`reopened`	REOPENED	The Buyer has confirmed that the Issue described in the Ticket has not been resolved satisfactorily and rejected the Seller's request to close the Ticket. The Ticket has been reopened and is waiting to continue being handled and investigated for resolution by the Seller.
`pending`	PENDING	The Seller is waiting on additional information in order to continue the handling of the Ticket. This may result in the clock being stopped for the service level agreement until the Buyer has responded to the request.

status	MEF 113 name	Description
assessingCancellation	ASSESSING_CANCELLATION	A request has been made by the Buyer to cancel the Ticket and is being assessed by the Seller to determine whether to just close the Ticket, or may also choose to resolve the Issue to prevent similar Create Ticket requests from other Buyers. If the Seller chooses to resolve the Issue, the Seller might create an Incident or an internal Ticket for the Issue, but that is outside the scope of this document. After the Seller has completed the assessment, the Seller updates the Ticket State to cancelled.
cancelled	CANCELLED	The Ticket has been successfully cancelled by the Buyer. This is a terminal state.

Table 7: Trouble Ticket statuses

[R14] The Seller **MUST** support all Trouble Ticket statuses and their associated transitions as described in Figure 8 and Table 7. [MEF113 R153]

[R15] The Buyer **MUST** set the source=buyer when adding any item to one of the following list: note, attachment, or relatedIssue. [MEF113 R11]

[R16] The Seller **MUST** set the source=seller when adding any item to one of the following list: note, attachment, or relatedIssue. [MEF113 R12]

[O3] The Seller **MAY** append an item to relatedContactInformation, note, attachment, relatedEntity, or relatedIssue if required. [MEF113 O7], [MEF113 O8], [MEF113 O9]

[O4] The Seller **MAY** add, modify, or delete an item in relatedContactInformation with role=sellerTechnicalContact. [MEF113 O9]

[O5] The Seller **MAY** add or modify an item in workOrder list. [MEF113 O11]

[R17] The Seller **MUST NOT** modify or delete any items provided by the Buyer in following lists: relatedContactInformation, note, attachment, relatedEntity, or relatedIssue. [MEF113 R2], [MEF113 R5], [MEF113 R27], [MEF113 R28], [MEF113 R29].

[R18] The Seller **MUST** add a note when any of the following Trouble Ticket attributes are updated: [MEF113 R26]

- `expectedResolutionDate`
- `relatedIssue`

6.2. Use Case 2: Retrieve Ticket List

[O6] The Buyer **MAY** retrieve a list of Trouble Tickets by using a `GET /troubleTicket` operation with desired filtering criteria. The attributes that are available to be used are: [MEF113 O12]

- `externalId`
- `priority`
- `sellerPriority`
- `severity`
- `sellerSeverity`
- `ticketType`
- `status`
- `relatedEntityId`
- `relatedEntityType`
- `creationDate.gt`
- `creationDate.lt`
- `expectedResolutionDate.gt`
- `expectedResolutionDate.lt`
- `resolutionDate.gt`
- `resolutionDate.lt`

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

```
https://serverRoot/mefApi/sonata/troubleTicket/v2/troubleTicket?
status=inProgress&priority=critical&limit=10&offset=0
```

The example above shows a Buyer's request to get all Trouble Tickets that are in the `inProgress` status and with `critical` priority. Additionally, the Buyer asks only for a first (`offset=0`) pack of 10 results (`limit=0`) to be returned. The correct response (HTTP code `200`) in the response body contains a list of `TroubleTicket_Find` objects matching the criteria. To get more details (e.g. the item level information), the Buyer has to query a specific `TroubleTicket` by `id`.

[R19] The Seller **MUST** put the following attributes (if set) into the `TroubleTicket_Find` object in the response: [MEF113 R32]:

- `id`
- `externalId`
- `relatedEntity`
- `description`

- `priority`
- `sellerPriority`
- `severity`
- `sellerSeverity`
- `ticketType`
- `status`
- `creationDate`
- `expectedResolutionDate`
- `resolutionDate`

[R20] In case no items matching the criteria are found, the Seller **MUST** return a valid response with an empty list. [MEF113 R32]

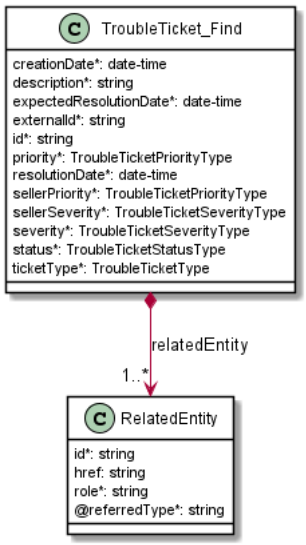


Figure 9: Use Case 2: Retrieve Ticket List - Model

6.3. Use Case 3: Retrieve Ticket by Ticket Identifier

The Buyer can get detailed information about the Trouble Ticket from the Seller by using a `GET /troubleTicket/{{id}}` operation.

[R21] In case `id` does not allow to find a `TroubleTicket` instance in Seller's system, an error response `Error404` **MUST** be returned. [MEF113 R35]

[R22] The Seller **MUST** put the following attributes into the `TroubleTicket` object in the response: [MEF113 R37]

- `id`
- `relatedEntity`
- `description`
- `priority`
- `sellerPriority`
- `severity`

- `sellerSeverity`
- `ticketType`
- `status`
- `creationDate`
- `relatedContactInformation` - items with `role` equal to `reporterContact` and `sellerTicketContact`

[R23] The Seller **MUST** provide all remaining optional attributes if they were previously set by the Buyer or the Seller. [MEF113 R38]

[R24] The Seller's response to a Retrieve Ticket by Ticket Identifier request **MUST** include the `resolutionDate` and a `note` added by the Seller describing how the Ticket was resolved if the `status` is `closed` or `resolved`. [MEF113 R39]

6.4. Use Case 4: Patch Ticket by Ticket Identifier

The update operation is realized with the use of the REST PATCH operation. For that purpose, a specialized type `TroubleTicket_Update` is provided. It consists of attributes limited to a subset that includes only the Buyer updateable attributes.

The PATCH usage recommendation follows TMF 621 json/merge (https://tools.ietf.org/html/rfc7386).

Figure 10 presents the model used in the PATCH request. The Seller responds with a `TroubleTicket` type.

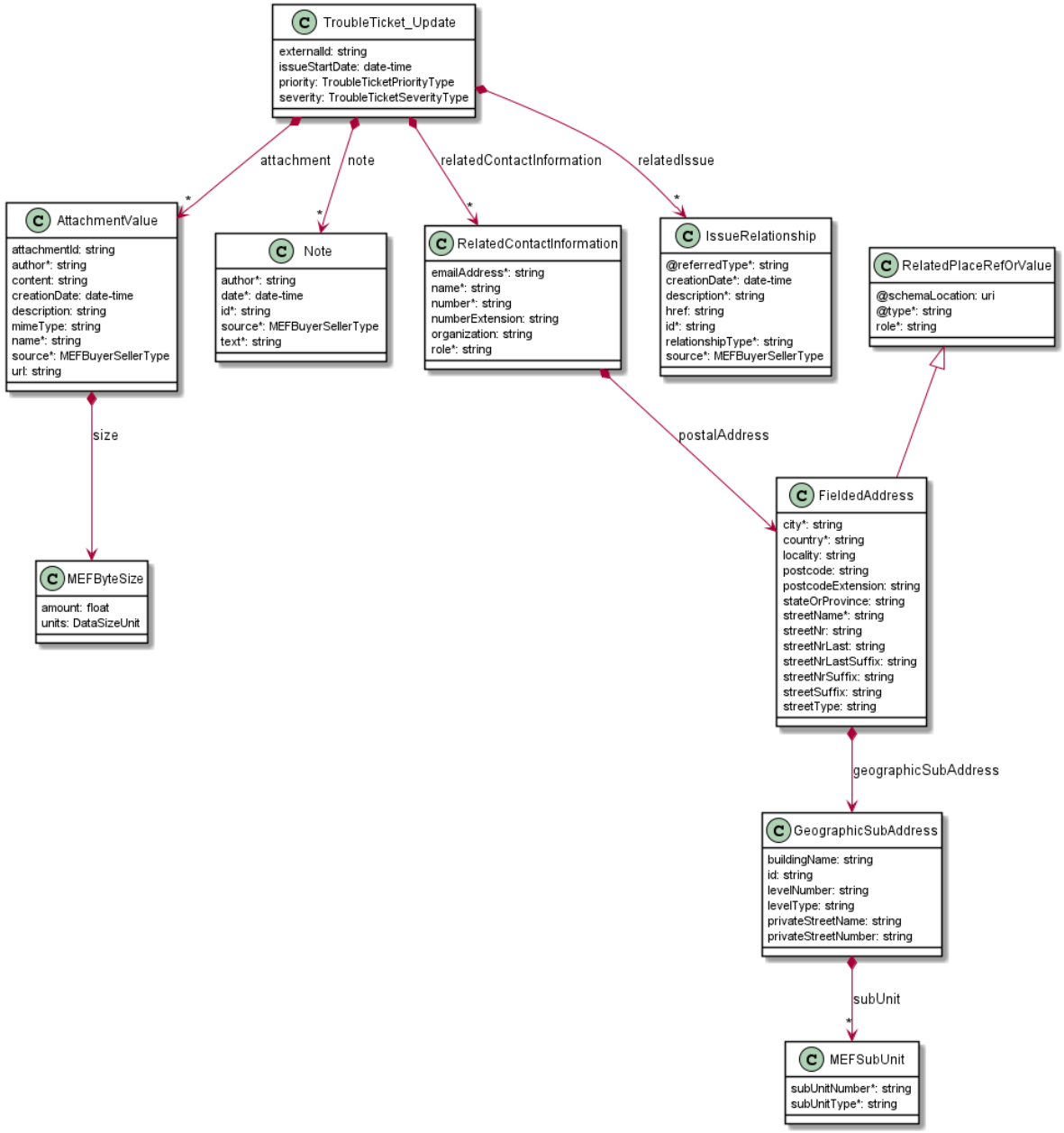


Figure 10: Patch request Model

[R25] The Buyer **MUST** include at least one of the following attributes of `TroubleTicket_Update` in the PATCH request: [MEF113 R41]

- `externalId`
- `priority`
- `severity`
- `issueStartDate`
- `note` - append only
- `relatedContactInformation` - append or modify the Buyer settable contacts
- `relatedIssue`

[R26] The Buyer **MUST** add a `note` to a Trouble Ticket when any of the following attributes are patched: [MEF113 R42]

- `priority`

- `severity`
- `issueStartDate`
- `relatedIssue`

[R27] The Buyer **MUST NOT** modify or delete any items provided by the Seller in following lists: `relatedContactInformation`, `attachment`. `note relatedEntity`, or `relatedIssue`. [MEF113 R3], [MEF113 R6], [MEF113 R44]

Note: The Buyer can add or update items in above-mentioned lists by providing a full list of existing items, and appending them with new ones or updating values of existing ones (where possible).

[R28] In case `id` does not allow to find a `TroubleTicket` that is to be updated in Seller's system, an error response `Error404` **MUST** be returned. [MEF113 R46]

[R29] The Seller **MUST** return an error (`Error422`) if attributes requested to be changed by the Buyer cannot be updated. [MEF113 R47]

[R30] The Seller **MUST** return an error (`Error422`) if the Ticket `state` is `closed`, `assessingCancellation` or `cancelled`. [MEF113 R48]

The example below shows a request to:

- add a new `note` (existing cannot be modified or deleted)
- change details of Buyer's `reporterContact`

```
{
  "note": [
    {<<previously existing>>
      "id": "note-1",
      "author": "John Example",
      "date": "2021-06-02T14:25:11.090Z",
      "source": "buyer",
      "text": "Couldn't reach the support on phone."
    },
    {<<added new note>>
      "id": "note-2",
      "author": "Kate Example",
      "date": "2021-06-02T19:25:11.090Z",
      "source": "buyer",
      "text": "Support reached after 5 hours"
    }
  ],
  "relatedContactInformation": [
    {<< update details of reporterContact >>
      "emailAddress": "Kate.example@example.com",
      "name": "Kate Example",
      "number": "+12-345-678-91",
      "organization": "Buyer Example Co.",
      "role": "reporterContact"
    },
    {<< provided by Buyer - untouched >>
      "emailAddress": "Seller.TicketContact@example.com",
      "name": "Seller Ticket Contact",
      "number": "+98-765-432-10",
      "organization": "Seller Example Co.",
      "role": "sellerTicketContact"
    }
  ]
}
```

[R31] If the Trouble Ticket status is `pending`, the Seller **MUST** update it to `inProgress`. [MEF113 R53]

6.5. Use case 5: Cancel Ticket by Ticket Identifier

The Buyer may request to Cancel a Trouble Ticket by using `POST /troubleTicket/{{id}}/cancel` endpoint. This operation only requires providing the `id` in the path and has an empty `204` confirmation response.

The sequence diagram below presents this use case in detail.

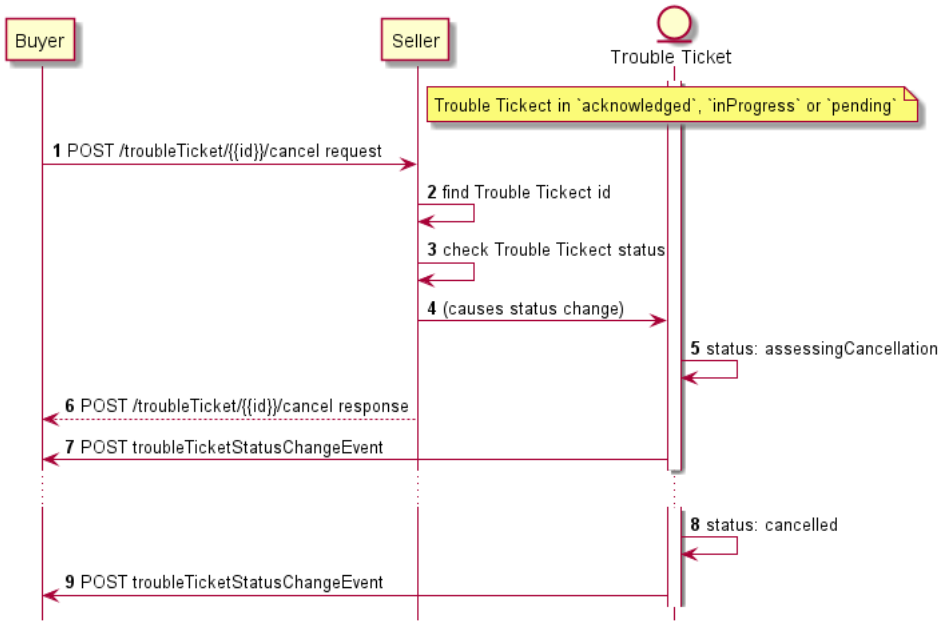


Figure 11: Cancel Trouble Ticket Flow

The Seller verifies the request, then searches for a Trouble Ticket to be cancelled by given `id`. If found, the status is verified (`acknowledged`, `inProgress` or `pending` allowed). I everything is verified correctly, the Seller moves the ticket to the `assessingCancellation` status, sends a successful response to a cancellation request, and starts assessing the cancellation process for the ticket.

[R32] In case of a successful validation the Seller **MUST** move the ticket to `assessingCancellation` status. [MEF113 R58]

[R33] In case `id` does not allow to find a `TroubleTicket` that is to be cancelled, an error response `Error404` **MUST** be returned. [MEF113 R56]

[R34] In case the `TroubleTicket` is in one of statuses `resolved`, `closed`, `reopened`, `assessingCancellation`, or `cancelled` Seller **MUST** return an error (`Error422`). [MEF113 R57]

6.6 Use Case 6: Respond to Ticket Clearance Notification

As shown in Figure 5, the Seller after resolving the Issue moves the Trouble Ticket to a `resolved` state. This is the point where the Buyer verifies the resolution and chooses to either close or reopen the Trouble Ticket. The Seller sends the `troubleTicketResolvedEvent` - a dedicated notification type to the Buyer. The Buyer uses one of the dedicated actions:

- `POST /troubleTicket/{{id}}/close`
- `POST /troubleTicket/{{id}}/reopen`

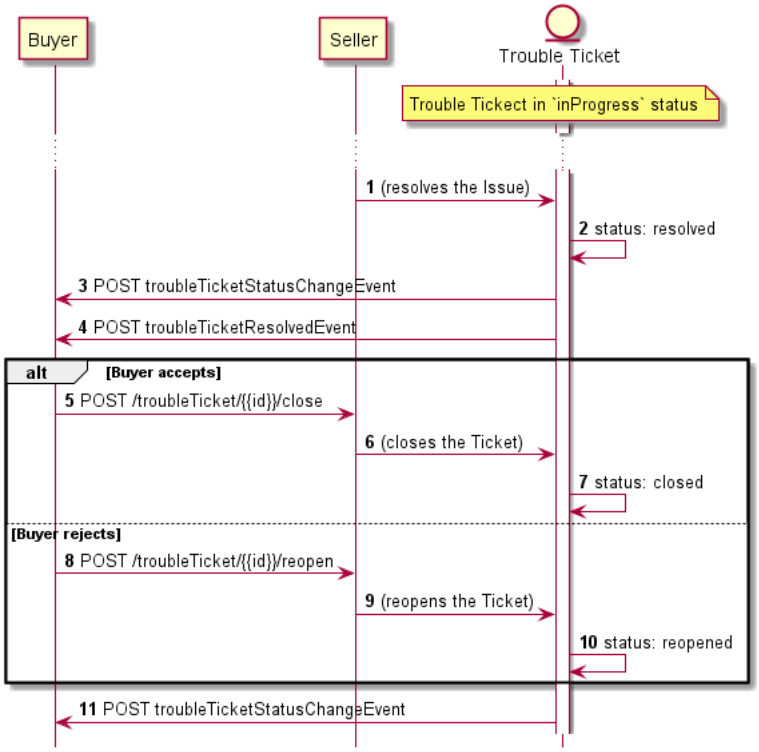


Figure 12: Respond to Ticket Clearance Notification Flow

[R35] The Buyer **MUST** perform the `reopen` action if the Issue on which the Ticket was based has not been resolved in a satisfactory manner to the Buyer. [MEF113 R60]

[R36] If performing the `reopen` action, the Buyer **MUST** include a `reason` describing why the Buyer doesn't agree that the Trouble Ticket has been resolved in a satisfactory manner and is requesting the Trouble Ticket to be reopened. [MEF113 R61]

[R37] The Buyer **MUST** perform the `close` action if the Issue on which the Ticket was based has been resolved in a satisfactory manner to the Buyer. [MEF113 R62]

[R38] In case `id` does not allow to find a `TroubleTicket` that is to be reopened or closed, an error response `Error404` **MUST** be returned. [MEF113 R64]

[R39] If Buyer performs the `reopen` action, the Seller **MUST** change the Ticket state to `reopened`. [MEF113 R65]

[R40] If Buyer performs the `close` action, the Seller **MUST** change the Ticket state to `closed`. [MEF113 R66]

6.7. Use Case 12: Retrieve Workorder by Workorder Identifier

The Buyer can retrieve detailed information about the Workorder from the Seller by using a `GET /workOrder/{{id}}` operation.

[R41] In case `id` does not allow to find a `Workorder` instance, an error response `Error404` **MUST** be returned. [MEF113 R111]

[R42] The Seller **MUST** put the following attributes into the `Workorder` object in the response: [MEF113 R113]

- `appointmentRequired`
- `id`
- `relatedContactInformation` - item with `role=technicalContact`
- `relatedEntity`
- `state`
- `task`

[R43] The Seller **MUST** provide all remaining optional attributes if they were previously set by the Buyer or the Seller. [MEF113 R114]

The following Figure presents the model of the WorkOrder.

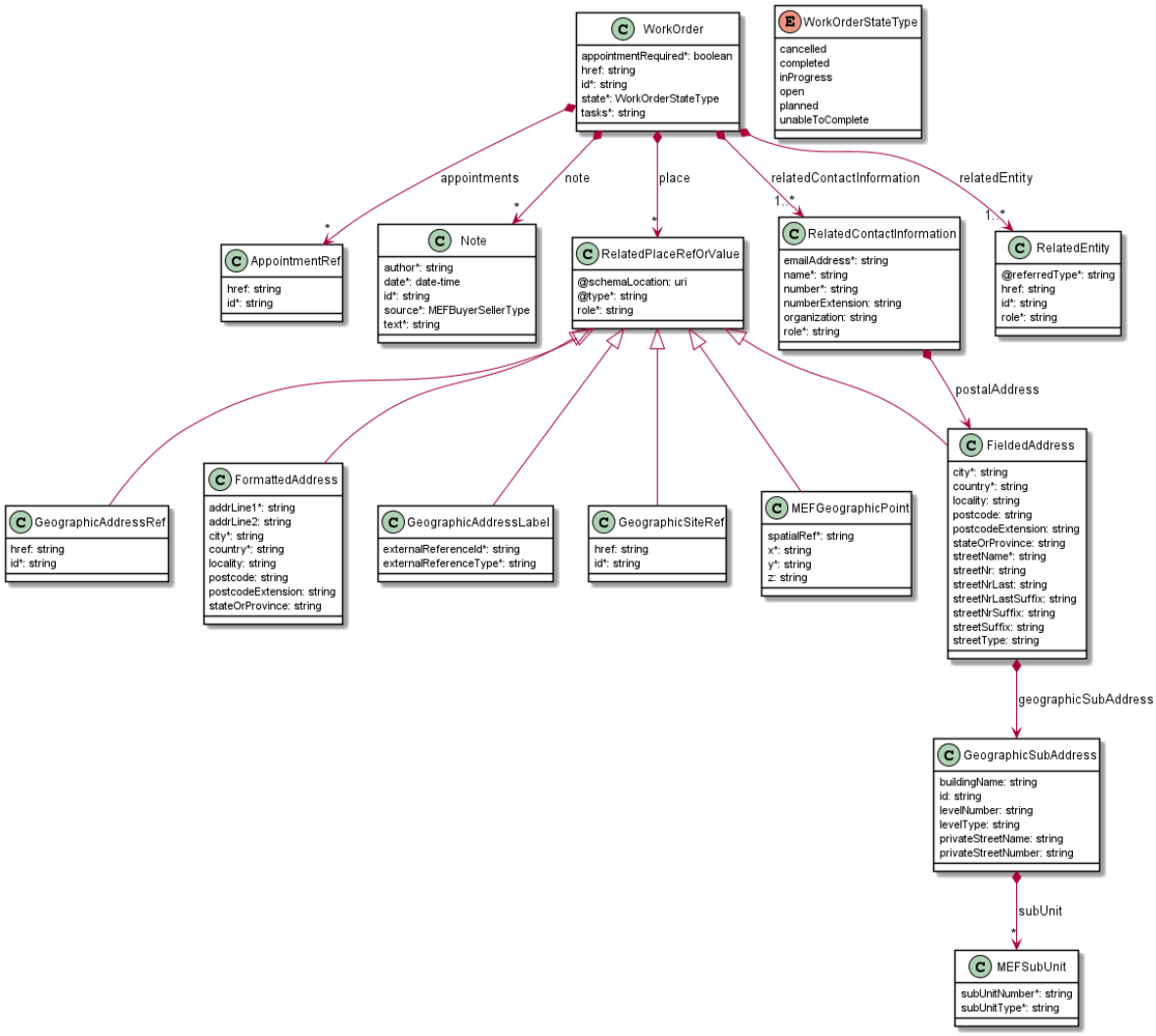


Figure 13: Use Case 12: Workorder - Model

Table 8 presents the mapping between the API `state` names and the MEF 113 naming, together with their description.

state	MEF 113 name	Description
`completed`	COMPLETED	The Seller Technician responsible for the Workorder has successfully completed all the assigned Tasks.
`cancelled`	CANCELLED	The Workorder has been cancelled by the Seller or due to the Buyer requesting to cancel the Ticket.
`inProgress`	IN_PROGRESS	The Seller Technician responsible for the Workorder has been assigned and started one or more of the assigned Tasks.
`open`	OPEN	A Workorder was initiated by the Seller to be assigned to a Technician responsible for resolving the Ticket.

state	MEF 113 name	Description
`planned`	PLANNED	The Workorder has been given an execution date for resolving one or more Tasks.
`unableToComplete`	UNABLE_TO_COMPLETE	The Seller Technician responsible for the Workorder was unable to complete one or more of the assigned Tasks, because of additional skills or information is required. Additional tasks are required to resolve the Ticket and a new Workorder needs to be opened.

Table 8: Workorder states

Figure 14 presents the WorkOrder state machine:

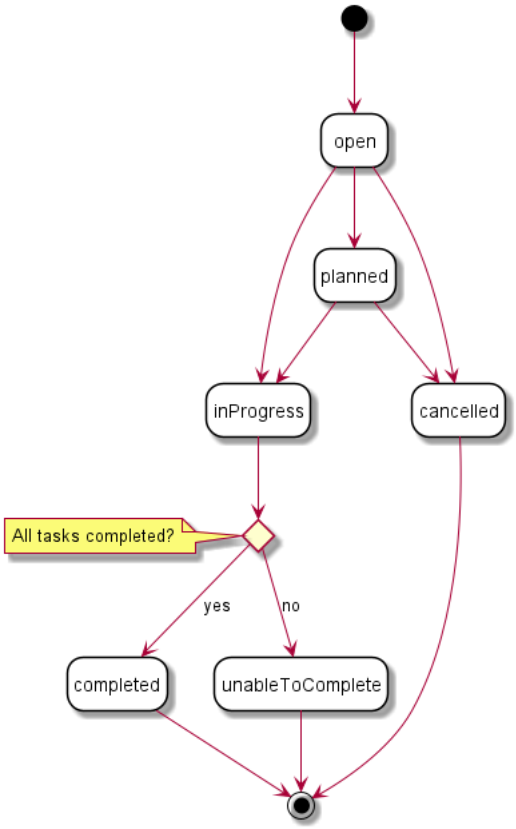


Figure 14: WorkOrder State Machine

[R44] The Seller **MUST** support all WorkOrder states and their associated transitions as described in Figure 14 and Table 8. [MEF113 R154]

Below you can find a snippet with a Workorder example:


```
{
  "id": "98765432-9876-5432-0000-000000000055",
  "appointmentRequired": false,
  "relatedContactInformation": [
    {
```

```
        "emailAddress": "technical.contact@example.com",
        "name": "Technical Contact",
        "number": "+12-345-678-91",
        "organization": "Seller Example Co.",
        "role": "technicalContact"
      }
    ],
    "relatedEntity": [
      {
        "id": "01494079-6c79-4a25-83f7-48284196d44d",
        "role": "Issue Source",
        "@referredType": "Product"
      }
    ],
    "state": "open",
    "task": ["Replace the broken SFP", "Perform OTDR"]
  }
```


6.8. Use Case 13: Retrieve Incident List

[O7] The Buyer **MAY** retrieve a list of Incidents by using a `GET /incident` operation with desired filtering criteria. The attributes that are available to be used are: [MEF113 O17]

- `priority`
- `severity`
- `incidentType`
- `status`
- `relatedEntityId`
- `relatedEntityType`
- `creationDate.gt`
- `creationDate.lt`
- `issueStartDate.gt`
- `issueStartDate.lt`
- `expectedClosedDate.gt`
- `expectedClosedDate.lt`
- `closedDate.gt`
- `closedDate.lt`

The example of making a request and using pagination is provided in section 6.2 Please refer to it as the rules also apply to this case.

[R45] The Seller **MUST** put the following attributes (if set) into the `Incident_Find` object in the response: [MEF113 R117]:

- `id`
- `relatedEntity`
- `description`
- `priority`
- `severity`
- `incidentType`
- `status`

- `creationDate`
- `issueStartDate`
- `expectedClosedDate`
- `closedDate`

[R46] In case no items matching the criteria are found, the Seller **MUST** return a valid response with an empty list. [MEF113 R117]

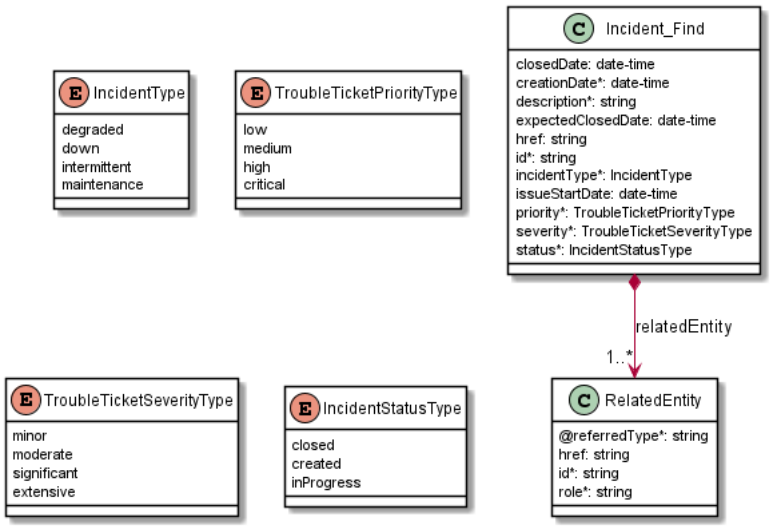


Figure 15: Use Case 13: Retrieve Incident List - Model

6.9. Use Case 14: Retrieve Incident by Incident Identifier

The Buyer can get detailed information about the Incident from the Seller by using a `GET /incident/{{id}}` operation.

[R47] In case `id` does not allow to find an `Incident` instance, an error response `Error404` **MUST** be returned. [MEF113 R120]

[R48] The Seller **MUST** put the following attributes into the `Incident` object in the response: [MEF113 R122]

- `id`
- `relatedEntity`
- `description`
- `priority`
- `severity`
- `incidentType`
- `status`
- `creationDate`
- `relatedContactInformation` - items with `role` equal to `incidentContact`

[R49] The Seller **MUST** provide all remaining optional attributes if they were previously set by the Buyer or the Seller. [MEF113 R123]

[R50] The Seller's response to a Retrieve Incident by Incident Identifier request **MUST** include the `closedDate` if the `status` is `closed`. [MEF113 R124]

Table 9 presents the mapping between the API `status` names and the MEF 113 naming, together with their description.

status	MEF 113 name	Description
`created`	CREATED	A new Incident has been created and allocated a unique `id`.
`inProgress`	IN_PROGRESS	The Incident is in the process of being handled by the Seller.
`closed`	CLOSED	The Situation described in the Incident was closed by the Seller. This is a terminal state.

Table 9: Incident states

Figure 16 presents the Incident state machine:

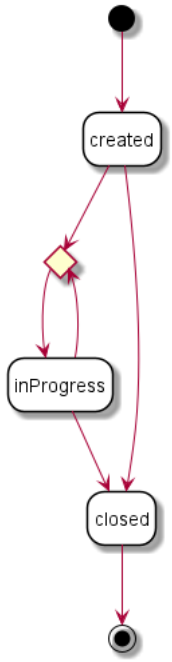


Figure 16: Incident State Machine

[R51] The Seller **MUST** support all Incident states and their associated transitions as described in Figure 16 and Table 9. [MEF113 R155]

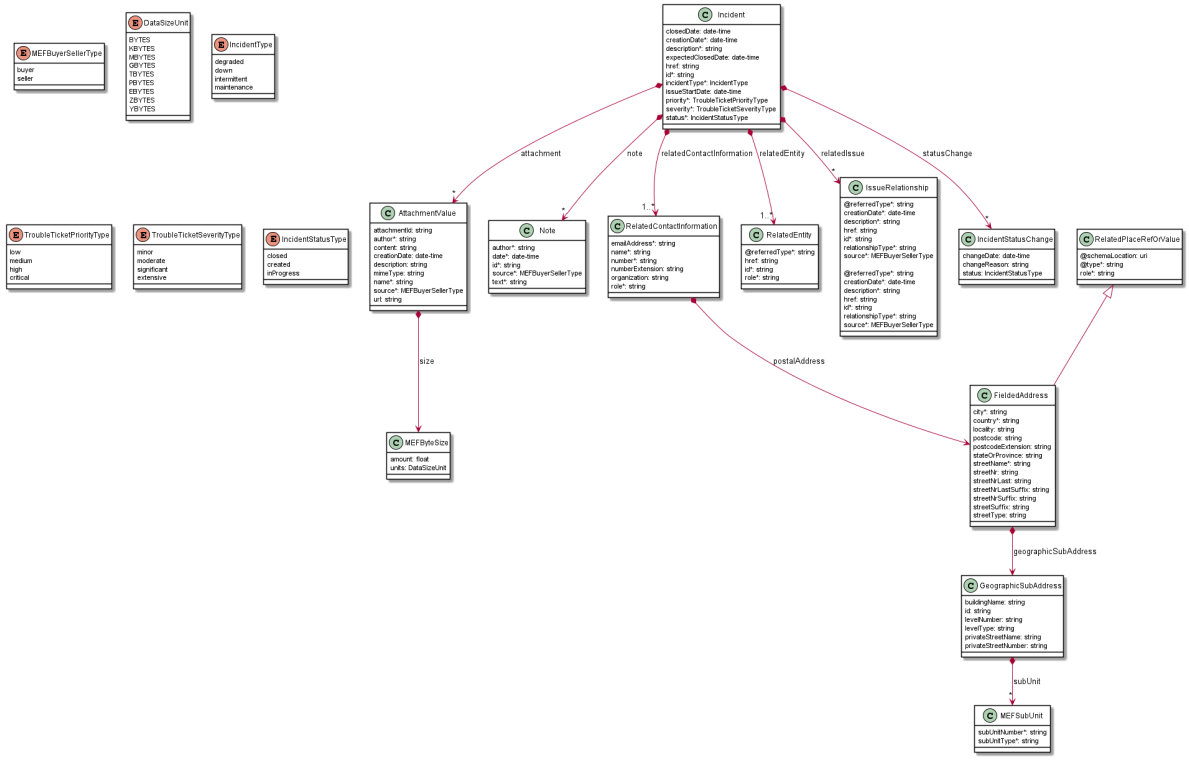


Figure 17: Use Case 14: Incident Model


```
{
  "id": "00001111-4321-6666-7777-000000003333",
  "href": "{{baseUrl}}/incident/00001111-4321-6666-7777-000000003333",
  "attachment": [
    {
      "attachmentId": "att-002",
      "author": "Kate Example",
      "creationDate": "2022-01-02T14:21:11.090Z",
      "description": "Print screen from the assurance system",
      "mimeType": "image/jpeg",
      "name": "Alarm",
      "url": "https://example.com/documents/00000000-5555-4444-3333-222211110000",
      "size": {
        "amount": 2.6,
        "units": "MBYTES"
      },
      "source": "seller"
    }
  ],
  "creationDate": "2022-01-12T23:09:44.814Z",
  "description": "Hardware failure",
  "expectedClosedDate": "2022-01-13T23:09:44.814Z",
  "incidentType": "down",
  "issueStartDate": "2022-01-12T23:09:44.814Z",
  "priority": "critical",
  "relatedContactInformation": [incidentContact
    {
      "emailAddress": "Incident.Contact@example.com",
      "name": "Incident Contact",
      "number": "+98-765-432-10",
      "organization": "Seller Example Co.",
      "role": "incidentContact"
    }
  ],
  "relatedEntity": [
    {
      "id": "01494079-6c79-4a25-83f7-48284196d44d",
      "role": "Affected Product",
      "@referredType": "Product"
    }
  ],
  "relatedIssue": [
    {
      "@referredType": "TroubleTicket",
      "creationDate": "2022-01-12T23:09:44.815Z",
```

```
      "description": "Reported failure is causing referred Trouble Ticket",
      "id": "00000000-4444-5555-6666-000000000987",
      "relationshipType": "causes",
      "source": "seller"
    }
  ],
  "severity": "extensive",
  "status": "created"
}
```

6.10. Use case 15: Register for Event Notifications

[R52] The Seller **MUST** support Trouble Ticket Notifications. [MEF113 R129]

[R53] The Buyer **MUST** support and register for Trouble Ticket Notifications. [MEF113 R130]

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

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

The usage of a combination of these attributes fulfills the [ME113 R125], [ME113 R126], [ME113 R127] requirements.

By using a simple request:

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

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

- `troubleTicketAttributeValueChangeEvent`
- `troubleTicketInformationRequiredEvent`
- `troubleTicketResolvedEvent`
- `troubleTicketStatusChangeEvent`
- `incidentCreatedEvent`
- `incidentAttributeValueChangeEvent`
- `incidentClosedEvent`
- `incidentStatusChangeEvent`

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

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

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

```
eventType=troubleTicketResolvedEvent,troubleTicketStatusChangeEvent
```

or

```
eventType=troubleTicketResolvedEvent&eventType=troubleTicketStatusChangeEvent
```

The `query` formatting complies to RCF3986 RFC3986. According to it, every attribute defined in the Event model (from notification API) can be used in the `query`. However, this standard requires only `eventType` attribute to be supported.

[R54] `eventType` is the only attribute that the Seller **MUST** support in the query.

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

```
{
  "id": "00000000-0000-0000-0000-000000000678",
  "callback": "https://buyer.com/listenerEndpoint",
  "query": "eventType=troubleTicketResolvedEvent"
}
```

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

- `https://buyer.com/listenerEndpoint/mefApi/sonata/troubleTicketNotification/v2/listener/troubleTicketResolvedEvent`

6.11. Use case 16: Send Ticket Notification

Notifications are used to asynchronously inform the Buyer about the respective objects and attributes changes. The Seller's synchronous response to a Trouble Ticket create requests are considered to act as a Create Notification so there is no explicit respective Create Notification type. The next notification must be sent when the state changes compared to the previously sent one.

[R55] The Seller **MUST** send Ticket Notifications to Buyers who have registered for them. [MEF113 R131]

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

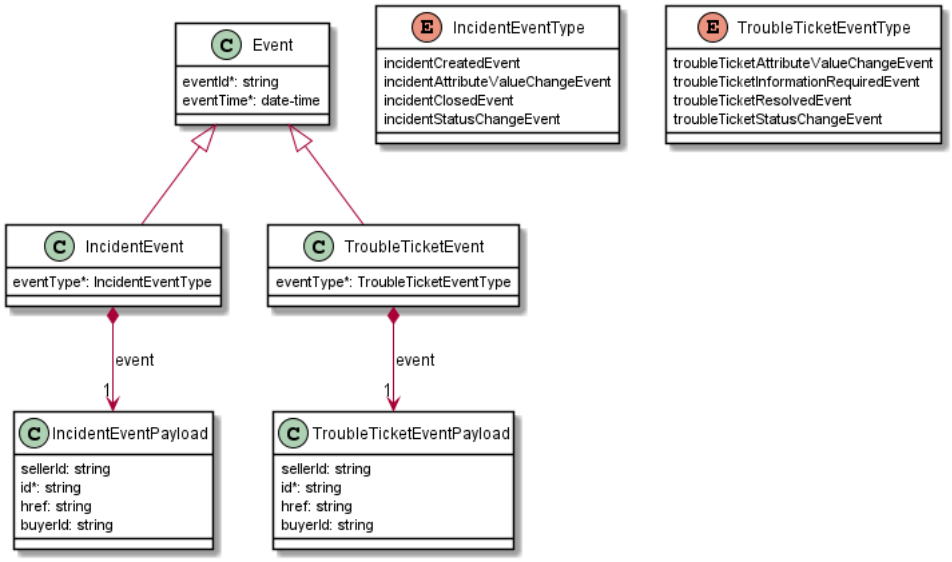


Figure 18: Use Case 15. Notification Data Model

The following snippet presents an example of `troubleTicketResolvedEvent`

```
{
  "eventId": "event-001",
  "eventType": "troubleTicketResolvedEvent",
  "eventTime": "2021-06-03T15:56:08.559Z",
  "event": {
    "id": "00000000-4444-5555-6666-000000000987"
  }
}
```

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

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

The table below presents the mapping between the API Notification types' names and the ones in MEF 113. The inconsistencies are caused by using the TMF event types as the base for this API.

API name	MEF 113 name
`troubleTicketAttributeValueChangeEvent`	UPDATE
`troubleTicketInformationRequiredEvent`	INFO_REQUIRED
`troubleTicketResolvedEvent`	CLEARANCE_REQUEST
`troubleTicketStatusChangeEvent`	STATE_CHANGE

Table 10. Trouble Ticket Notification types mapping

[R56] The Seller **MUST** send a `troubleTicketAttributeValueChangeEvent` whenever the Seller updates any of the following Ticket attributes: [MEF113 R133]

- `sellerSeverity`
- `sellerPriority`
- `expectedResolutionDate`
- `note`
- `attachment`
- `relatedContactInformation` - items with `role` equal to `sellerTicketContact` and `sellerTechnicalContact`
- `relatedIssue`
- `workOrder` - including updates to a Referenced Workorder

[R57] The Seller **MUST** send a `troubleTicketStatusChangeEvent` whenever a Ticket `status` change occurs. [MEF113 R124]

[R58] The Seller **MUST** set the Ticket's `status` to `pending` and add a `note` to the Ticket prior to sending the `troubleTicketInformationRequiredEvent` to inform the Buyer about what additional information is required to continue processing the Ticket. [MEF113 R135]

[R59] The Buyer **MUST** use the Patch Ticket by Ticket Identifier request to provide the missing information before the Seller is able to continue processing the Ticket. [MEF113 R137]

[R60] The Seller **MUST** send a `troubleTicketResolvedEvent` before closing an open Ticket. [MEF113 R138]

6.12. Use case 17: Send Incident Notification

[R61] The Seller **MUST NOT** send Incidents Notifications to Buyers who have not registered for them. [MEF113 R139]

[R62] The Seller **MUST** send Incidents Notifications to Buyers who have registered for them. [MEF113 R140]

The following snippet presents an example of `incidentClosedEvent`

```
{
  "eventId": "event-011",
  "eventType": "incidentClosedEvent",
  "eventTime": "2021-12-22T01:56:08.559Z",
  "event": {
    "id": "00000000-4444-5555-6666-000001110654"
  }
}
```

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

registration.

The table below presents the mapping between the API Notification types' names and the ones in MEF 113. The inconsistencies are caused by using the TMF event types as the base for this API.

API name	MEF 113 name
`incidentCreatedEvent`	CREATED
`incidentAttributeValueChangeEvent`	UPDATE
`incidentStatusChangeEvent`	STATE_CHANGE
`incidentClosedEvent`	CLOSED

Table 11. Incidents Notification types mapping

[R63] The Seller **MUST** send a `incidentCreatedEvent` whenever an Incident has been created. [MEF113 R142]

[R64] The Seller **MUST** send a `incidentAttributeValueChangeEvent` whenever the Seller updates any of the following Incident attributes: [MEF113 R144]

- `priority`
- `severity`
- `incidentType`
- `issueStartDate`
- `expectedClosedDate`
- `note`
- `attachment`
- `relatedContactInformation`
- `relatedIssue`
- `description`

[R65] The Seller **MUST** send a `incidentStatusChangeEvent` whenever an Incident `status` change occurs. [MEF113 R143]

[R66] The Seller **MUST** send a `incidentClosedEvent` whenever an Incident `status` changes to `closed`. [MEF113 R145]

7. API Details

7.1. API patterns

7.1.1. Indicating errors

Erroneous situations are indicated by appropriate HTTP responses. An error response is indicated by HTTP status 4xx (for client errors) or 5xx (for server errors) and appropriate response payload. The Product Order API uses the error responses as depicted and described below.

Implementations can use HTTP error codes not specified in this standard in compliance with rules defined in RFC 7231 [RFC7231]. In such a case, the error message body structure might be aligned with the `Error`.

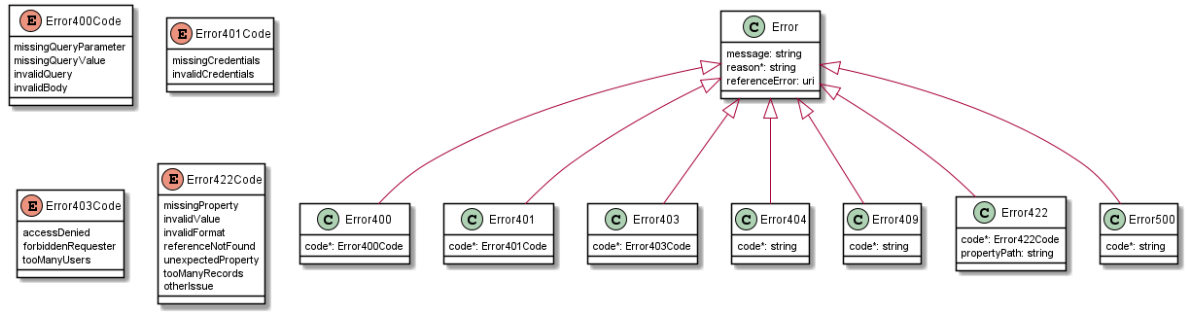


Figure 19. Data model types to represent an erroneous response

7.1.1.1. Type Error

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

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

7.1.1.2. Type Error400

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

Inherits from:

- Error

Name	Type	Description

Name	Type	Description
code*	Error400Code	One of the following error codes: - missingQueryParameter: The URI is missing a required query-string parameter - missingQueryValue: The URI is missing a required query-string parameter value - invalidQuery: The query section of the URI is invalid. - invalidBody: The request has an invalid body

7.1.1.3. enum Error400Code

Description: One of the following error codes:

- missingQueryParameter: The URI is missing a required query-string parameter
- missingQueryValue: The URI is missing a required query-string parameter value
- invalidQuery: The query section of the URI is invalid.
- invalidBody: The request has an invalid body

7.1.1.4. Type Error401

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

Inherits from:

- Error

Name	Type	Description
code*	Error401Code	One of the following error codes: - missingCredentials: No credentials provided. - invalidCredentials: Provided credentials are invalid or expired

7.1.1.5. enum Error401Code

Description: One of the following error codes:

- missingCredentials: No credentials provided.
- invalidCredentials: Provided credentials are invalid or expired

7.1.1.6. Type Error403

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

Inherits from:

- Error

Name	Type	Description
code*	Error403Code	This code indicates that the server understood the request but refuses to authorize it because of one of the following error codes: - accessDenied: Access denied - forbiddenRequester: Forbidden requester - tooManyUsers: Too many users

7.1.1.7. enum Error403Code

Description: This code indicates that the server understood the request but refuses to authorize it because of one of the following error codes:

- accessDenied: Access denied
- forbiddenRequester: Forbidden requester
- tooManyUsers: Too many users

7.1.1.8. Type Error404

Description: Resource for the requested path not found. (https://tools.ietf.org/html/rfc7231#section-6.5.4)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - notFound: A current representation for the target resource not found

7.1.1.9. Type Error409

Description: Conflict (https://datatracker.ietf.org/doc/html/rfc7231#section-6.5.8)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - conflict: The client has provided a value whose semantics are not appropriate for the property.

7.1.1.10. Type Error422

The response for HTTP status `422` is a list of elements that are structured using the `Error422` data type. Each list item describes a business validation problem. This type introduces the `propertyPath` attribute which points to the erroneous property of the request, so that the Buyer may fix it easier. It is highly recommended that this property should be used, yet remains optional because it might be hard to implement.

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

Inherits from:

- Error

Name	Type	Description
code*	Error422Code	One of the following error codes: - missingProperty: The property the Seller has expected is not present in the payload - invalidValue: The property has an incorrect value - invalidFormat: The property value does not comply with the expected value format - referenceNotFound: The object referenced by the property cannot be identified in the Seller system - unexpectedProperty: Additional property, not expected by the Seller has been provided - tooManyRecords: the number of records to be provided in the response exceeds the Seller's threshold. - otherIssue: Other problem was identified (detailed information provided in a reason)
propertyPath	string	A pointer to a particular property of the payload that caused the validation issue. It is highly recommended that this property should be used. Defined using JavaScript Object Notation (JSON) Pointer (https://tools.ietf.org/html/rfc6901).

7.1.1.11. `enum` Error422Code

Description: One of the following error codes:

- missingProperty: The property the Seller has expected is not present in the payload
- invalidValue: The property has an incorrect value
- invalidFormat: The property value does not comply with the expected value format

- referenceNotFound: The object referenced by the property cannot be identified in the Seller system
- unexpectedProperty: Additional property, not expected by the Seller has been provided
- tooManyRecords: the number of records to be provided in the response exceeds the Seller's threshold.
- otherIssue: Other problem was identified (detailed information provided in a reason)

7.1.1.12. Type Error500

Description: Internal Server Error. (https://tools.ietf.org/html/rfc7231#section-6.6.1)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - internalError: Internal server error - the server encountered an unexpected condition that prevented it from fulfilling the request.

7.1.1.13. Type Error501

Description: Not Implemented. Used in case Seller is not supporting an optional operation (https://tools.ietf.org/html/rfc7231#section-6.6.2)

Inherits from:

- Error

Name	Type	Description
code*	string	The following error code: - notImplemented: Method not supported by the server

7.1.2. Response pagination

A response to retrieve a list of results (e.g. `GET /productOfferingQualification`) can be paginated. The Buyer can specify following query attributes related to pagination:

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

The Seller returns a list of elements that comply with the requested `limit`. If the requested `limit` is higher than the supported list size the smaller list result is returned. In that case, the size of the result is returned in the header attribute `X-Result-Count`. The Seller can indicate that there are additional results available using:

- `X-Total-Count` header attribute with the total number of available results
- `X-Pagination-Throttled` header set to `true`

[R67] Seller **MUST** use either `X-Total-Count` or `X-Pagination-Throttled` to indicate that the page was truncated and additional results are available.

7.2. Management API Data model

Figure 20 presents the whole Trouble Ticket Management data model the data types, requirements related to them and mapping to MEF 113 specifications are discussed later in this section.

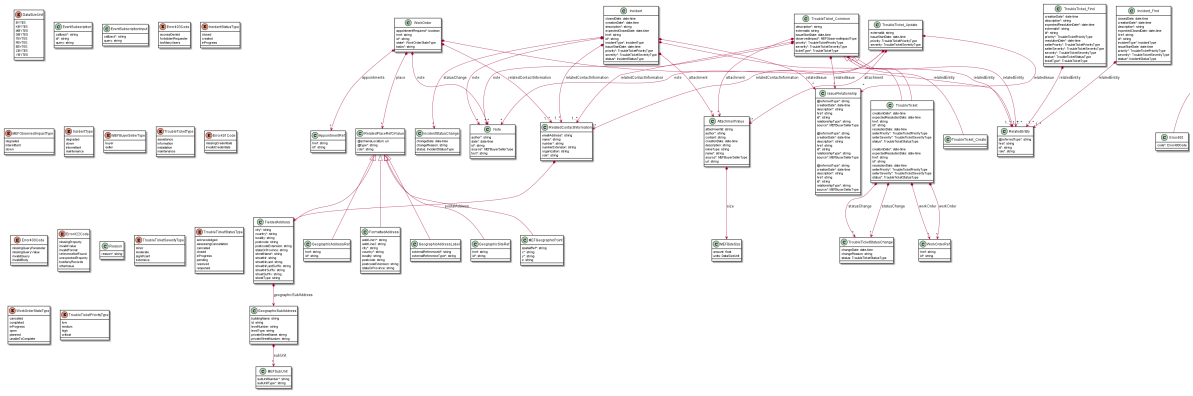


Figure 20: Trouble Ticket Management Data Model

7.2.1. TroubleTicket

7.2.1.1. Type TroubleTicket_Common

Description: A Trouble Ticket is a record of an issue that is created, tracked, and managed by a Trouble Ticket management system Skipped properties: id,href

Name	Type	Description	MEF 11
attachment	AttachmentValue[]	Attachments to the Ticket, such as a file, screen shot or embedded content. Attachments may be added but may not be modified or deleted (for historical reasons).	Attachm
description*	string	Summarized description of the Issue the Buyer is experiencing.	Descript

Name	Type	Description	MEF 11
externalId	string	Identifier provided by the Buyer to allow the Buyer to use as a search attribute in Retrieve Ticket List.	Buyer Ticket Identifie
issueStartDate	date-time	The date indicating when the Buyer first observed the Issue, to provide the Seller with additional insight.	Issue Sta Date
note	Note[]	The note(s) that are associated to the ticket. Notes may be added but may not be modified or deleted (for historical reasons).	Notes
observedImpact*	MEFObservedImpactType	The type of impact observed by the Buyer.	Observe Impact
priority*	TroubleTicketPriorityType	The priority of the Trouble Ticket and how quickly the issue should be resolved. Example: Critical, High, Medium, Low. The value is set by the ticket management system considering the severity, ticket type etc...	Priority
relatedContactInformation*	RelatedContactInformation[]	Party playing a role for this Trouble Ticket. The 'role' is to specify the type of contact as specified in MEF 113: Reporter Contact ('role=reporterContact') - REQUIRED in the request Buyer Technical Contacts ('role=buyerTechnicalContact') Seller Ticket Contact ('role=sellerTicketContact') Seller Technical Contact ('role=sellerTechnicalContact')	Reporter Contact, Buyer Technica Contacts Seller Ticket Contact, Seller Technica Contacts

Name	Type	Description	MEF 11
relatedEntity*	RelatedEntity[]	An entity that is related to the ticket such as a bill, a product, etc. The entity against which the ticket is associated.	Product Identifie
relatedIssue	IssueRelationship[]	A list of Related Issue relationships. Represents relationships to other Trouble Tickets and Incidents.	Related Objects
severity*	TroubleTicketSeverityType	The severity or impact (ITIL) of the Trouble Ticket as evaluated by the Buyer.	Severity
ticketType*	TroubleTicketType	The presumed cause of the Trouble Ticket as evaluated by the Buyer.	Type

7.2.1.2. Type TroubleTicket_Create

Description: A Trouble Ticket is a record of an issue that is created, tracked, and managed by a Trouble Ticket management system The modeling pattern introduces the `Common` supertype to aggregate attributes that are common to both `TroubleTicket` and `TroubleTicket_Create`. It this case the Create type has a subset of attributes of the response type and does not introduce any new, thus the `Create` type has an empty definition.

Inherits from:

- TroubleTicket_Common

7.2.1.3. Type TroubleTicket

Description: A Trouble Ticket is a record of an issue that is created, tracked, and managed by a Trouble Ticket management system

Inherits from:

- TroubleTicket_Common

Name	Type	Description	MEF 113
creationDate*	date-time	The date on which the Trouble Ticket was created	Ticket Creation Date

Name	Type	Description	MEF 113
expectedResolutionDate	date-time	The date provided by the Seller to indicate when the Ticket is expected to be resolved	Target Resolved Date
href	string	Hyperlink, a reference to the Trouble Ticket entity	Not represented in MEF 113
id*	string	Unique (within the Seller Ticket domain) identifier for the Ticket.	Seller Ticket Identifier
resolutionDate	date-time	The date the Ticket status was set to resolved by the Seller	Resolved Date
sellerPriority*	TroubleTicketPriorityType	The priority (ITIL) is based on the assessment of the impact and urgency of how quickly the Ticket should be resolved after evaluation by the Seller of the impact of the Issue on the Buyer.	Seller Priority
sellerSeverity*	TroubleTicketSeverityType	The severity or impact (ITIL) of the Ticket on the Buyer as evaluated by the Seller.	Seller Severity
status*	TroubleTicketStatusType	The current status of the Trouble Ticket	Ticket State

Name	Type	Description	MEF 113
statusChange	TroubleTicketStatusChange[]	The status change history that is associated to the ticket. Populated by the Seller.	Not represented in MEF 113
workOrder	WorkOrderRef[]	A reference to a set of WorkOrders to be performed under the responsibility of Seller technician(s) to resolve the Ticket.	Workorders

7.2.1.4. Type TroubleTicket_Find

Description: This class represents a single list item for the response of `listTroubleTicket` operation.

Name	Type	Description	MEF 113
creationDate*	date-time	The date on which the Trouble Ticket was created	Ticket Creation Date
description*	string	Description of the trouble or issue	Description
expectedResolutionDate*	date-time	The date provided by the Seller to indicate when the Ticket is expected to be resolved	Target Resolved Date
externalId*	string	Additional identifier coming from an external system	Buyer Ticket Identifier
id*	string	Unique identifier of the Trouble Ticket	Seller Ticket Identifier

Name	Type	Description	MEF 113
priority*	TroubleTicketPriorityType	The priority (ITIL) is based on the assessment of the impact and urgency of how quickly the Ticket should be resolved as evaluated by the Buy-er. The Priority is used by the Seller to determine the order in which Tickets get resolved across Buyers.	Priority
relatedEntity*	RelatedEntity[]	An entity that is related to the ticket such as a bill, a product, etc. The entity against which the ticket is associated.	Product Identifier
resolutionDate*	date-time	The date the Ticket status was set to resolved by the Seller	Resolved Date
sellerPriority*	TroubleTicketPriorityType	The priority (ITIL) is based on the assessment of the impact and urgency of how quickly the Ticket should be resolved after evaluation by the Seller of the impact of the Issue on the Buyer.	Seller Priority

Name	Type	Description	MEF 113
sellerSeverity*	TroubleTicketSeverityType	The severity or impact (ITIL) of the Ticket on the Buyer as evaluated by the Seller.	Seller Severity
severity*	TroubleTicketSeverityType	The severity or impact (ITIL) of the Ticket as evaluated by the Buyer.	Severity
status*	TroubleTicketStatusType	The current status of the Trouble Ticket	Not represented in MEF 113
ticketType*	TroubleTicketType	The presumed cause of the Trouble Ticket as evaluated by the Buyer.	Type

7.2.1.5. Type TroubleTicket_Update

Description: A Trouble Ticket is a record of an issue that is created, tracked, and managed by a Trouble Ticket management system

Name	Type	Description	MEF 1
attachment	AttachmentValue[]	Attachments to the Ticket, such as a file, screen shot or embedded content.	Attachn
externalId	string	Additional identifier coming from an external system	Buyer T Identifi
issueStartDate	date-time	The date indicating when the Buyer first observed the Issue, to provide the Seller with additional insight.	issueSta
note	Note[]	The note(s) that are associated to the ticket.	Notes

Name	Type	Description	MEF 1
priority	TroubleTicketPriorityType	The priority of the Trouble Ticket and how quickly the issue should be resolved. Example: Critical, High, Medium, Low. The value is set by the ticket management system considering the severity, ticket type etc...	Priority
relatedContactInformation	RelatedContactInformation[]	Party playing a role for this quote. If `instantSyncQuote=false` then the Buyer MUST specify Buyer Contact Information ('role=buyerContactInformation') and the Seller MUST specify Seller Contact Information ('role=sellerContactInformation')	Reporte Contact Buyer Technic Contact Seller T Contact Seller Technic Contact
relatedIssue	IssueRelationship[]	A list of Related Issue relationships. Represents relationships to other Trouble Tickets and Incidents.	Related Objects
severity	TroubleTicketSeverityType	The severity of the issue. Indicate the implication of the issue on the expected functionality e.g. of a system, application, service etc..	Not represe MEF 11

7.2.1.6. enum TroubleTicketPriorityType

Description: Possible values for the priority of the Trouble Ticket

Value
low
medium
high
critical

7.2.1.7. Type IssueRelationship

Description: Represents relationships to other Trouble Tickets and Incidents

Name	Type	Description	MEF 113
@referredType*	string	The actual type of the target instance when needed for disambiguation (Incident or TroubleTicket)	Related Object Type
creationDate*	date-time	The date the Relationship was created	Relation Date
description*	string	A description of the reason for the Relationship	Relation Description
href	string	Reference of the Trouble Ticket or Incident	Not represented in MEF 113
id*	string	Unique identifier of the referenced Issue (Trouble Ticket od Incident)	Related Object Identifier
relationshipType*	string	Type of the Trouble Ticket relationship can be blocks, depends on, duplicates, causes, etc...	Relation Type
source*	MEFBuyerSellerType	Indicates if this Related Issue was added by the Buyer or the Seller.	Relation Source

7.2.1.8. enum TroubleTicketSeverityType

Description: Possible values for the severity of the Trouble Ticket

Value
minor
moderate
significant
extensive

7.2.1.9. enum MEFObservedImpactType

Description: An enumeration of the possible values of impact observed by the Buyer.

- degraded: When the Product is impacted and not meeting the Product specifications.
- intermittent: When the Product is not operational as intended on an intermittent basis.
- down: When the Product is non-operational.

Value
degraded
intermittent
down

7.2.1.10. Type TroubleTicketStatusChange

Description: Holds the status notification reasons and associated date the status changed, populated by the server

Name	Type	Description	MEF 113
changeDate	date-time	The date and time the status changed.	Not represented in MEF 113
changeReason	string	The reason why the status changed.	Not represented in MEF 113
status	TroubleTicketStatusType	Reached state	Not represented in MEF 113

7.2.1.11. `enum` TroubleTicketStatusType

Description: Possible values for the status of the Trouble Ticket

status	MEF 113 name	Description
`acknowledged`	ACKNOWLEDGED	A request to create a Ticket was received and accepted by the Seller. The Ticket has been validated and created by the Seller and allocated a unique `id`.
`inProgress`	IN_PROGRESS	The Ticket is in the process of being handled and investigated for resolution by the Seller.

status	MEF 113 name	Description
`resolved`	RESOLVED	The Buyer's Issue described in the Ticket was resolved by the Seller. The Seller is now waiting for the Buyer to confirm that the Issue they reported is no longer observed.
`closed`	CLOSED	The Buyer that created the Ticket has confirmed that the Issue they reported is no longer observed, or the pre-defined time frame (agreed upon between Buyer and Seller) for confirming that the Issue has been resolved has passed without a response by the Buyer. This is a terminal state.
`reopened`	REOPENED	The Buyer has confirmed that the Issue described in the Ticket has not been resolved satisfactorily and rejected the Seller's request to close the Ticket. The Ticket has been reopened and is waiting to continue being handled and investigated for resolution by the Seller.
`pending`	PENDING	The Seller is waiting on additional information in order to continue the handling of the Ticket. This may result in the clock being stopped for the service level agreement until the Buyer has responded to the request.

status	MEF 113 name	Description
assessingCancellation	ASSESSING_CANCELLATION	A request has been made by the Buyer to cancel the Ticket and is being assessed by the Seller to determine whether to just close the Ticket, or may also choose to resolve the Issue to prevent similar Create Ticket requests from other Buyers. If the Seller chooses to resolve the Issue, the Seller might create an Incident or an internal Ticket for the Issue, but that is outside the scope of this document. After the Seller has completed the assessment, the Seller updates the Ticket State to `cancelled`.
cancelled	CANCELLED	The Ticket has been successfully cancelled by the Buyer. This is a terminal state.

7.2.1.12. `enum` TroubleTicketType

Description: Possible values for the type of the Trouble Ticket:

- assistance: Requesting help for a situation (not a failure) requiring attention that is not categorized.
- information: Buyer is requesting information on the Issue
- installation: Related to installation issue. Provisioning is complete, but Product is not operational.
- maintenance: Any scheduled or non-scheduled maintenance related Issue.

Value	MEF 113
assistance	ASSISTANCE
information	INFORMATION
installation	INSTALLATION
maintenance	MAINTENANCE

7.2.1.13. Type Reason

Description: An object to convey a reason for the operation.

Name	Type	Description	MEF 113
reason*	string	A text description of why given operation was requested.	Closure Rejection Reason

7.2.2. Incident

7.2.2.1. Type Incident

Description: An Incident is a record of an issue that is not part of normal operation in the Seller's network that has a possible negative impact on the operability of the network on one or more Buyers.

Name	Type	Description	MEF
attachment	AttachmentValue[]	Attachments to the Ticket, such as a file, screen shot or embedded content. Attachments may be added but may not be modified or deleted (for historical reasons).	Attach
closedDate	date-time	The date the Incident status was set to closed by the Seller	Incide Closec
creationDate*	date-time	The date on which the Incident was created	Incide Creati Date
description*	string	Description of the Incident	Descri
expectedClosedDate	date-time	The date provided by the Seller to indicate when the Incident is expected to be closed.	Incide Expec Closec
href	string	Hyperlink, a reference to the Incident entity	Not repres in ME
id*	string	Unique (within the Seller domain) identifier for the Incident.	Incide Identi
incidentType*	IncidentType	The presumed cause of the Incident as evaluated by the Seller.	Incide Type

Name	Type	Description	MEF
issueStartDate	date-time	The date when the Incident was first identified, for example via error logs.	Incide Start I
note	Note[]	A set of unstructured comments or information associated to the Incident. Notes may be added but may not be modified or deleted (for historical reasons).	Incide Notes
priority*	TroubleTicketPriorityType	The priority (ITIL) is based on the assessment of the impact and urgency of how quickly the Incident should be resolved after evaluation by the Seller of the impact of the Incident.	Incide Priorit
relatedContactInformation*	RelatedContactInformation[]	Party playing a role for this Incident. The 'role' is to specify the type of contact as specified in MEF 113: Incident Contact ('role=incidentContact') - REQUIRED to be set by the Seller Incident Technical Contact ('role=incidentTechnicalContact')	Incide Contac Incide Techni Contac
relatedEntity*	RelatedEntity[]	An entity that is related to the Incident such as a service, a product, etc. The entity which the Incident is associated with.	Produc Identif
relatedIssue	IssueRelationship[]	A list of Related Issue relationships. Represents relationships to other Trouble Tickets and Incidents.	Relate Object
severity*	TroubleTicketSeverityType	The severity or impact (ITIL) of the Incident as evaluated by the Seller.	Incide Severi
status*	IncidentStatusType	The current status of the Incident	Incide State
statusChange	IncidentStatusChange[]	The status change history that is associated to the Incident. Populated by the Seller.	Not repres in ME

7.2.2.2. Type Incident_Find

Description: This class represents a single list item for the response of `listIncident` operation.

Name	Type	Description	MEF 113
closedDate	date-time	The date the Incident status was set to closed by the Seller	Incident Closed Date
creationDate*	date-time	The date on which the Incident was created	Incident Creation Date
description*	string	Description of the Incident	Description
expectedClosedDate	date-time	The date provided by the Seller to indicate when the Incident is expected to be closed.	Incident Expected Closed Date
href	string	Hyperlink, a reference to the Incident entity	Not represented in MEF 113
id*	string	Unique (within the Seller domain) identifier for the Incident.	Incident Identifier
incidentType*	IncidentType	The presumed cause of the Incident as evaluated by the Seller.	Incident Type
issueStartDate	date-time	The date when the Incident was first identified, for example via error logs.	Incident Start Date
priority*	TroubleTicketPriorityType	The priority (ITIL) is based on the assessment of the impact and urgency of how quickly the Incident should be resolved after evaluation by the Seller of the impact of the Incident.	Incident Priority

Name	Type	Description	MEF 113
relatedEntity*	RelatedEntity[]	An entity that is related to the Incident such as a service, a product, etc. The entity which the Incident is associated with.	Product Identifier
severity*	TroubleTicketSeverityType	The severity or impact (ITIL) of the Incident as evaluated by the Seller.	Incident Severity
status*	IncidentStatusType	The current status of the Incident	Incident State

7.2.2.3. enum IncidentType

Description: Possible values for the type of the Incident:

- maintenance: Any scheduled or non-scheduled maintenance related Incident.
- degraded: When the Product is impacted and not meeting the Product specifications.
- intermittent: When the Product is not operational as intended on an intermittent basis
- down: When the Product is non-operational.

Value	MEF 113
degraded	DEGRADED
down	DOWN
intermittent	INTERMITTENT
maintenance	MAINTENANCE

7.2.2.4. enum IncidentStatusType

Description: Possible values for the status of the Incident

status	MEF 113 name	Description
created	CREATED	A new Incident has been created and allocated a unique id.
inProgress	IN_PROGRESS	The Incident is in the process of being handled by the Seller.
closed	CLOSED	The Situation described in the Incident was closed by the Seller. This is a terminal state.

7.2.2.5. Type IncidentStatusChange

Description: Holds the status notification reasons and associated date the status changed, populated by the server

Name	Type	Description	MEF 113
changeDate	date-time	The date and time the status changed.	Not represented in MEF 113
changeReason	string	The reason why the status changed.	Not represented in MEF 113
status	IncidentStatusType	Reached state	Not represented in MEF 113

7.2.3. Workorder

7.2.3.1. Type WorkOrder

Description: A set of tasks to be scheduled and performed under the responsibility of a Seller Technician(s)

Name	Type	Description	MEF 113
appointmentRequired*	boolean	The Seller requires the Buyer to schedule an Appointment. If set to true, the Seller is Requesting the Buyer to schedule an Appointment.	Appointment Required
appointment	AppointmentRef[]	A reference to a set of Appointments for the WorkOrder. A WorkOrder may contain only one open Appointment at a time (e.g. with state of 'scheduled').	Workorder Appointments
href	string	Hyperlink, a reference to the WorkOrder entity	Not represented in MEF 113
id*	string	Unique (within the Seller domain) identifier for the WorkOrder.	Workorder Identifier

Name	Type	Description	MEF 113
note	Note[]	A set of unstructured comments or information associated to the WorkOrder	Workorder Notes
place	RelatedPlaceRefOrValue[]	The location where the WorkOrder Tasks are to be performed. If an appointment is needed, this will also be the location where the Appointment takes place and includes the site contact which the Seller technician may need to get access to the Buyer's site during the Appointment. This could be an end-user, security personnel or any authorized person	Workorder Place

Name	Type	Description	MEF 113
relatedContactInformation*	RelatedContactInformation[]	Party playing a role for this WorkOrder. The 'role' is to specify the type of contact as specified in MEF 113: Technical Contact ('role=technicalContact') - REQUIRED to be set by the Seller. The Seller technical contact responsible for the WorkOrder. Technician ('role=technician') - The Seller technician assigned to the WorkOrder and responsible for performing a set of tasks. In certain instances this could be a Buyer technician that works on behalf of the Seller.	Technical Contact, Technician
relatedEntity*	RelatedEntity[]	An entity that is related to the WorkOrder such as a service, a product, etc. The entity which the WorkOrder is associated with.	Workorder Related Entity
state*	WorkOrderStateType	The current status of the WorkOrder	Workorder State

Name	Type	Description	MEF 113
task	string[]	A set of tasks to be performed under the responsibility of the Technician to fulfil the WorkOrder. Each item is a description of a specific task to be performed under the responsibility of the Technician.	Tasks

7.2.3.2. enum WorkOrderStateType

Description: Possible values for the status of the WorkOrder

state	MEF 113 name	Description
completed	COMPLETED	The Seller Technician responsible for the Workorder has successfully completed all the assigned Tasks.
cancelled	CANCELLED	The WorkOrder has been cancelled by the Seller or due to the Buyer requesting to cancel the Ticket.
inProgress	IN_PROGRESS	The Seller Technician responsible for the Workorder has been assigned and started one or more of the assigned Tasks.
open	OPEN	A WorkOrder was initiated by the Seller to be assigned to a Technician responsible for resolving the Ticket.
planned	PLANNED	The WorkOrder has been given an execution date for resolving one or more Tasks.
unableToComplete	UNABLE_TO_COMPLETE	The Seller Technician responsible for the Workorder was unable to complete one or more of the assigned Tasks, because additional skills or information is required. Additional tasks are required to resolve the Ticket and a new Workorder needs to be opened.

7.2.3.3. Type WorkOrderRef

Description: A reference to an WorkOrder resource.

Name	Type	Description	MEF 113
href	string	Hyperlink to the referenced WorkOrder.	Not represented in MEF 113
id*	string	Identifier of the referenced WorkOrder.	Workorder Identifier

7.2.4. Common

Types described in this subsection are shared among two or more Cantata and Sonata APIs.

7.2.4.1. Type AppointmentRef

Description: A reference to an Appointment resource available through Appointment API.

Name	Type	Description	MEF 113
href	string	Hyperlink to the referenced Appointment. Hyperlink MAY be used by the Seller in responses. Hyperlink MUST be ignored by the Seller in case it is provided by the Buyer in a request	Not represented in MEF 113
id*	string	Identifier of the referenced Appointment.	Appointment Identifier

7.2.4.2. Type AttachmentValue

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

Name	Type	Description	MEF 113
attachmentId	string	locally unique identifier to distinguish items from the Attachment list.	Not represented in MEF 113
author*	string	Author of the Attachment	Attachment Author
content	string	The actual contents of the attachment object, if embedded, encoded as base64. Either url or (content and mimeType) attributes MUST be provided during creation.	Content

Name	Type	Description	MEF 113
creationDate	date-time	The date the Attachment was added.	Attachment Date
description	string	A narrative text describing the content of the attachment	Description
mimeType	string	Attachment mime type such as extension file for video, picture and document	Mime Type
name*	string	The name of the attachment	Attachment Name
size	MEFByteSize	The size of the attachment.	Size
source*	MEFBuyerSellerType	Indicates if the attachment was added by the Buyer or the Seller.	Attachment Source
url	string	URL where the attachment is located. Either url or (content and mimeType) attributes MUST be provided during creation.	URL

7.2.4.3. enum DataSizeUnit

Description: The unit of measure in the data size.

Value
BYTES
KBYTES
MBYTES
GBYTES
TBYTES
PBYTES
EBYTES
ZBYTES
YBYTES

7.2.4.4. Type FieldedAddress

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

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 113
city*	string	The city that the address is in	City
country*	string	Country that the address is in	Country
geographicSubAddress	GeographicSubAddress	Additional fields used to specify an address, as detailed as possible.	Not represented in MEF 57.2
locality	string	The locality that the address is in	Locality
postcode	string	Descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as zip code)	Postal Code
postcodeExtension	string	An extension of a postal code. E.g. the part following the dash in a US urban property address	Postal Code Extension
stateOrProvince	string	The State or Province that the address is in	State Or Province
streetName*	string	Name of the street or other street type	Street Name
streetNr	string	Number identifying a specific property on a public street. It may be combined with streetNrLast for ranged addresses. MEF 79 defines it as required however as in certain countries it is not used we make it optional in API.	Street Number

Name	Type	Description	MEF 113
streetNrLast	string	Last number in a range of street numbers allocated to a property	Street Number Last
streetNrLastSuffix	string	Last street number suffix for a ranged address	Street Number Suffix Last
streetNrSuffix	string	The first street number suffix	Street Number Suffix
streetSuffix	string	A modifier denoting a relative direction	Street Suffix
streetType	string	The type of street (e.g., alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf)	Street Type

7.2.4.5. Type FormattedAddress

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

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 113
addrLine1*	string	The first address line in a formatted address	Address Line 1
addrLine2	string	The second address line in a formatted address	Address Line 2
city*	string	The city that the address is in	City
country*	string	Country that the address is in	Country
locality	string	An area of defined or undefined boundaries within a local authority or other legislatively defined area, usually rural or semi-rural in nature	Locality

Name	Type	Description	MEF 113
postcode	string	Descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also known as ZIP code)	Postal Code
postcodeExtension	string	An extension of a postal code. E.g. the part following the dash in an US urban property address	Postal Code Extension
stateOrProvince	string	The State or Province that the address is in	State Or Province

7.2.4.6. Type GeographicAddressLabel

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

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 113
externalReferenceId*	string	A reference to an address by id	Administrative Authority Address Label
externalReferenceType*	string	Uniquely identifies the authority that specifies the addresses reference and/or its type (if the authority specifies more than one type of address). The value(s) to be used are to be agreed during the onboarding. For North American providers this would normally be CLLI (Common Language Location Identifier) code.	Administrative Authority

7.2.4.7. Type GeographicAddressRef

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

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 113
href	string	Hyperlink to the referenced GeographicAddress. Hyperlink MAY be used by the Seller in responses. Hyperlink MUST be ignored by the Seller in case it is provided by the Buyer in a request	Not represented in MEF 113
id*	string	Identifier of the referenced Geographic Address. This identifier is assigned during a successful address validation request (Geographic Address Validation API)	Not represented in MEF 113

7.2.4.8. Type GeographicSiteRef

ERROR -> Unrecognized type: 'AppoinGeographicSiteReftmentRef'!

7.2.4.9. Type GeographicSubAddress

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

Name	Type	Description	MEF 113
buildingName	string	Allows for identification of places that require building name as part of addressing information	Building Name
id	string	Unique Identifier of the subAddress	Not represented in MEF 113
levelNumber	string	Used where a level type may be repeated e.g. BASEMENT 1, BASEMENT 2	Level Number
levelType	string	Describes level types within a building	Level Type
privateStreetName	string	"Private streets internal to a property (e.g. a university) may have internal names that are not recorded by the land title office	Private Street Name
privateStreetNumber	string	Private streets numbers internal to a private street	Private Street Number

Name	Type	Description	MEF 113
subUnit	MEFSubUnit[]	Representation of a MEFSubUnit It is used for describing subunit within a subaddress e.g.BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.	Not represented in MEF 80

7.2.4.10. enum MEFBuyerSellerType

Description: An enumeration with buyer and seller values.

Value	MEF 113
buyer	BUYER
seller	SELLER

7.2.4.11. Type MEFByteSize

Description: A size represented by value and Byte units

Name	Type	Description	MEF 113
amount	float	Numeric value in a given unit	Value
units	DataSizeUnit	Byte Unit	Unit

7.2.4.12. Type MEFGeographicPoint

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

Inherits from:

- RelatedPlaceRefOrValue

Name	Type	Description	MEF 113
spatialRef*	string	The spatial reference system used to determine the coordinates (e.g. "WGS84"). The system used and the value of this field are to be agreed during the onboarding process.	Spatial Reference
x*	string	The latitude expressed in the format specified by the `spacialRef`	Latitude
y*	string	The longitude expressed in the format specified by the `spacialRef`	Longitude

Name	Type	Description	MEF 113
z	string	The elevation expressed in the format specified by the `spacialRef`	Elevation

7.2.4.13. Type MEFSubUnit

Description: Allows for sub unit identification

Name	Type	Description	MEF 113
subUnitNumber*	string	The discriminator used for the subunit, often just a simple number but may also be a range.	Sub Unit Name
subUnitType*	string	The type of subunit e.g.BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF.	Sub Unit Type

7.2.4.14. Type Note

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

Name	Type	Description	MEF 113
author*	string	Author of the note	Note Author
date*	date-time	Date of the note	Note Date
id*	string	Identifier of the note within its containing entity (may or may not be globally unique, depending on provider implementation)	Not represented in MEF 113
source*	MEFBuyerSellerType	Indicates if this Note was added by the Buyer or Seller.	Note Source
text*	string	Text of the note	Note Text

7.2.4.15. Type RelatedContactInformation

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

Name	Type	Description	MEF 113

Name	Type	Description	MEF 113
emailAddress*	string	Email address	Contact email Address
name*	string	Name of the contact	Contact Name
number*	string	Phone number	Contract Phone Number
numberExtension	string	Phone number extension	Contract Phone Number Extension
organization	string	The organization or company that the contact belongs to	Contact Organization
postalAddress	FieldedAddress	Identifies the postal address of the person or office to be contacted.	Contact Postal Address
role*	string	A role the party plays in a given context.	Not represented in MEF 80

7.2.4.16. Type RelatedEntity

Description: A reference to an entity, where the type of the entity is not known in advance.

Name	Type	Description	MEF 113
@referredType*	string	The actual type of the target instance when needed for disambiguation.	Not represented in MEF 113
href	string	Reference of the related entity.	Not represented in MEF 113
id*	string	Unique identifier of a related entity.	Product Identifier
role*	string	The role of an entity.	Not represented in MEF 113

7.2.4.17. Type RelatedPlaceRefOrValue

Description: Defines the Place (Address or Site) by reference or by value.

Name	Type	Description	MEF 113

Name	Type	Description	MEF 113
@schemaLocation	uri	A URI to a JSON-Schema file that defines additional attributes and relationships. May be used to define additional related place types. Usage of this attribute must be agreed upon between Buyer and Seller.	Not represented in MEF 113
@type*	string	This field is used as a discriminator and is used between different place representations. This type might discriminate for additional related place as defined in '@schemaLocation'.	Not represented in MEF 113
role*	string	Role of this place	RelatedPlaceRefOrValue

7.2.5. Notification registration

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

7.2.5.1. Type EventSubscriptionInput

Description: This class is used to register for Notifications.

Name	Type	Description
callback*	string	This callback value must be set to *host* property from Buyer Notification API (trc This property is appended with the base path and notification resource path specifie which notification is sent. E.g. for "callback": "http://buyer.com/listenerEndpoint", will be sent to: `http://buyer.com/listenerEndpoint/mefApi/sonata/troubleTicketNotification/v2/liste
query	string	This attribute is used to define to which type of events to register to. Example: "que troubleTicketStatusChangeEvent". To subscribe for more than one event type, put tl `eventType=troubleTicketStatusChangeEvent,troubleTicketResolvedEvent`. The pc 'TroubleTicketEventType' in troubleTicketNotification.api.yaml. An empty query is ending in subscription for all event types.

7.2.5.2. Type EventSubscription

Description: Sets the communication endpoint address the service instance must use to deliver notification information

Name	Type	Description	MEF 113
callback*	string	The value provided by the Buyer in `EventSubscriptionInput` during notification registration	Notification Target Information
id*	string	An identifier of the event subscription assigned by the Seller when a resource is created.	Not represented in MEF 113
query	string	This attribute is used to define notification registration constraints.	List of Notification Event Types, Action

7.3. Notification API Data model

Figure 21 presents the Trouble Ticket Management Notification data model.

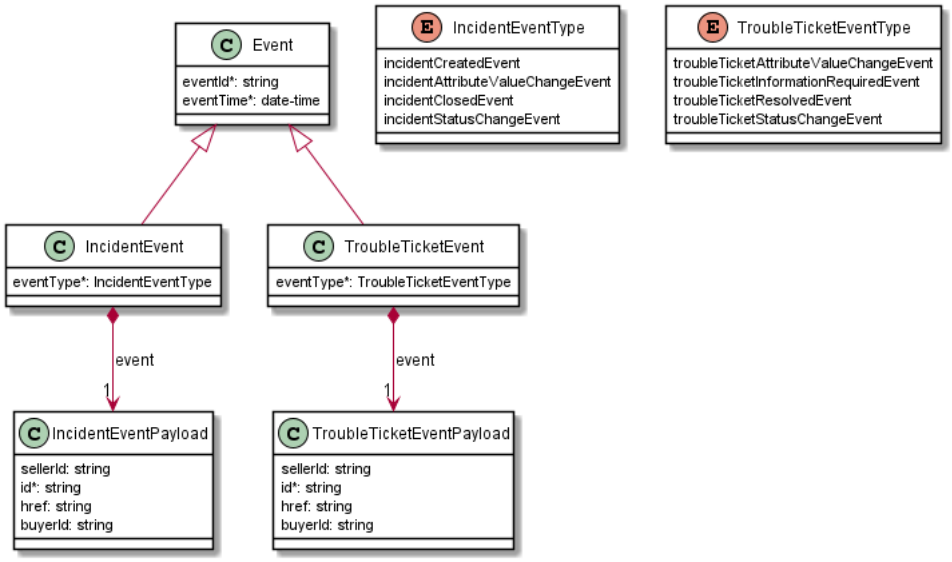


Figure 21. Trouble Ticket Management Notification Data Model

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

7.3.1. Type Event

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

Name	Type	Description	MEF 113
eventId*	string	Id of the event	Not represented in MEF 113
eventTime*	date-time	Datetime when the event occurred	Not represented in MEF 113

7.3.2. Type TroubleTicketEvent

Description:

Inherits from:

- Event

Name	Type	Description	MEF 113
eventType*	TroubleTicketEventType	Indicates the type of the event.	Notification Type
event*	TroubleTicketEventPayload	A reference to the object that is source of the notification.	Not represented in MEF 113

7.3.3. enum TroubleTicketEventType

Description: Type of the Trouble Ticket Event

Value	MEF 113
troubleTicketAttributeValueChangeEvent	UPDATE
troubleTicketInformationRequiredEvent	INFO_REQUIRED
troubleTicketResolvedEvent	CLEARANCE_REQUEST
troubleTicketStatusChangeEvent	STATE_CHANGE

7.3.4. Type TroubleTicketEventPayload

Description: The identifier of the Trouble Ticket being subject of this event.

Name	Type	Description	MEF 113
sellerId	string	The unique identifier of the organization that is acting as the Seller. MUST be specified in the request only when requester entity represents more than one Seller.	Seller
id*	string	ID of the Trouble Ticket attributed by quoting system	Not represented in MEF 113
href	string	Hyperlink to access the Trouble Ticket	Not represented in MEF 113
buyerId	string	The unique identifier of the organization that is acting as the a Buyer. MUST be specified in the request only when the responding represents more than one Buyer.	Buyer

7.3.5. Type IncidentEvent

Description:

Inherits from:

- Event

Name	Type	Description	MEF 113
eventType*	IncidentEventType	Indicates the type of the event.	Notification Type
event*	IncidentEventPayload	A reference to the object that is source of the notification.	Not represented in MEF 113

7.3.6. Type IncidentEventPayload

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

Name	Type	Description	MEF 113
sellerId	string	The unique identifier of the organization that is acting as the Seller. MUST be specified in the request only when requester entity represents more than one Seller.	Seller
id*	string	ID of the Incident attributed by quoting system	Not represented in MEF 113
href	string	Hyperlink to access the Incident	Not represented in MEF 113
buyerId	string	The unique identifier of the organization that is acting as the a Buyer. MUST be specified in the request only when the responding represents more than one Buyer.	Buyer

7.3.7. enum IncidentEventType

Description: Type of the Incident Event

Value	MEF 113
incidentCreatedEvent	CREATED
incidentAttributeValueChangeEvent	UPDATE

Value	MEF 113
incidentClosedEvent	CLOSED
incidentStatusChangeEvent	STATE_CHANGE

8. References

- [OAS-v3] Open API 3.0, February 2020
- [MEF55.1] MEF 55.1, Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, February 2021
- [MEF79] MEF 79, Address, Service Site, and Product Offering Qualification Management, Requirements and Use Cases, November 2019
- [MEF80] MEF 80, Quote Management Requirements and Use Cases, July 2021
- [MEF113] MEF W113 0.15Trouble Ticketing Business Requirements and Use Cases, January 2022, Draft Standard (R2)
- [REST] Chapter 5: Representational State Transfer (REST) Fielding, Roy Thomas, Architectural Styles and the Design of Network-based Software Architectures (Ph.D.).
- [RFC2119] RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, March 1997
- [RFC3986] RFC 3986 Uniform Resource Identifier (URI): Generic Syntax, January 2005
- [RFC8174] RFC 8174, Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, May 2017
- [TMF630] TMF 630 TMF630 API Design Guidelines 4.2.0
- [TMF621] TMF 621, Trouble Ticket API REST Specification R19.0.1, November 2019