Open Industry 4.0 Alliance

Development Guideline for Open Edge Computing

Version: 1.0.0
File:Open Industry 4.0 Alliance - Development Guideline
Editor (Authors listed in A4):

Name	Email	Company
Konrad Heidrich	kheidrich@hilscher.com	Hilscher Gesellschaft für Systemautomation mbH

OPEN INDUSTRY 4.0 ALLIANCE

Content

List of figures

List of tables

Preface

The *Open Edge Computing - Development Guide* is a document which is the result of the activities within the Open Edge Computing Working Group of the Open Industry 4.0 Alliance. The goal of this document is to provide a starting point for developers and architects on developing edge computing applications or devices that are compliant to the Open Industry 4.0 Alliance requirements and architecture. As this document is a guideline and not a specification, some contents are still ambiguous or in discussion as well as some of the definitions given may be subject to change.

Main focus of this document is the description of the message exchange between the Field, Edge and Cloud layers of the Open Industry 4.0 Alliance overall architecture. The messages defined within the Open Edge Computing layer are based on existing industry standards such as OPC-UA and MQTT. In the first chapters, this document introduces the basic concepts, models and paradigms necessary to build the fundaments of Open Edge Computing. Chapter 3 and 4 present two essential concepts of Open Edge Computing layer: the oi4Identifier and the Master Asset Model (mam). Chapter 5 details processes and scenarios that describes the interactions of the components within the edge computing layer with the underlying layer (field devices and assets) as well as overlying layer (Cloud). While Chapters 6 elaborates requirements and settings for Docker Containers, Chapter 7 elaborates those for MQTT brokers and clients to be Open Industry 4.0 Alliance-compliant. In the following chapter 8, the corresponding MQTT topics are introduced. Chapter 9 gives a detailed introduction of the message payload format based on OPC UA-conform JSON format to be used within the edge computing layer. It also includes the descriptions of OPC-UA objects and methods as defined by the Open Industry 4.0 Alliance. Chapter 10 details the Message Bus communications based on the definitions of the previous sections. Finally, chapter 11 introduces the requirements of Open Edge Comping Platform regarding components consisted within the architecture, hardware, applications, and software licenses.

In the short time that the technical committee of the Open Industry 4.0 Alliance has worked, we achieved quick success in interoperability thanks to an open and result-oriented mindset as well as significant resource commitment of all founder partners. I cordially invite all new members of the Open Industry 4.0 Alliance to join in this spirit of entrepreneurial collaboration, to shape a tangible, truly interoperable Industry 4.0.

Many thanks to all who participated in the workshops, shared their knowledge and did their homework afterwards. Thank you for your commitment and ideas that led us to this guideline and closer to the vision of an interoperable and value-added industry 4.0.

Hans-Jürgen Hilscher,
Lead of the Technical Committee of the Open Industry 4.0 Alliance

1 Conventions

The following symbols are used for highlighting important information and recommendations.

- **NOTE** for common hints
  Example: "*If no local CA is present, the global CA (see 6.1.2) is used.*"
- **RECOMMENDATION** normal recommendation
  Example: "*To prevent access by unauthorized communication participants, the operator must prevent this via ACL.*"
- **ADVISORY** strong recommendation
  Example: "*Do not store any private keys here!*"
- **ATTENTION** *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

In case code or command line input / output are used this is done using a dedicated style with line numbering.

Example:

```
PS C:\User> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
fbfa578081ad   bridge    bridge    local
10b78e0e3c3a   host      host      local
37c2dffba462   none      null      local
PS C:\User>
```

References to Elements used in messages or topics are highlighted in gray background.

Example:

`MessageType`

For the illustration of Interactions UML sequence diagrams are used.

2 Introduction to Open Edge Computing - Considered Standards and Technical Decisions

NOTE *The Open Industry 4.0 Alliance is committed to use existing technologies wherever they are feasible, and complement them with best practices and specifications to achieve true interoperability for Industry 4.0 solutions.*

As shown in the Figure below, the reference architecture of the Open Industry 4.0 Alliance introduce four layers: Open Edge Connectivity, Open Edge Computing, Open Operator Cloud Platform, and Common Cloud Central. This document mainly focuses on defining and describing the Open Edge Computing layer at its core, but also considering its interactions with Open Edge Connectivity, Open Operator Cloud Platform, and Common Cloud Central layers. These layers are discussed in separate documents.

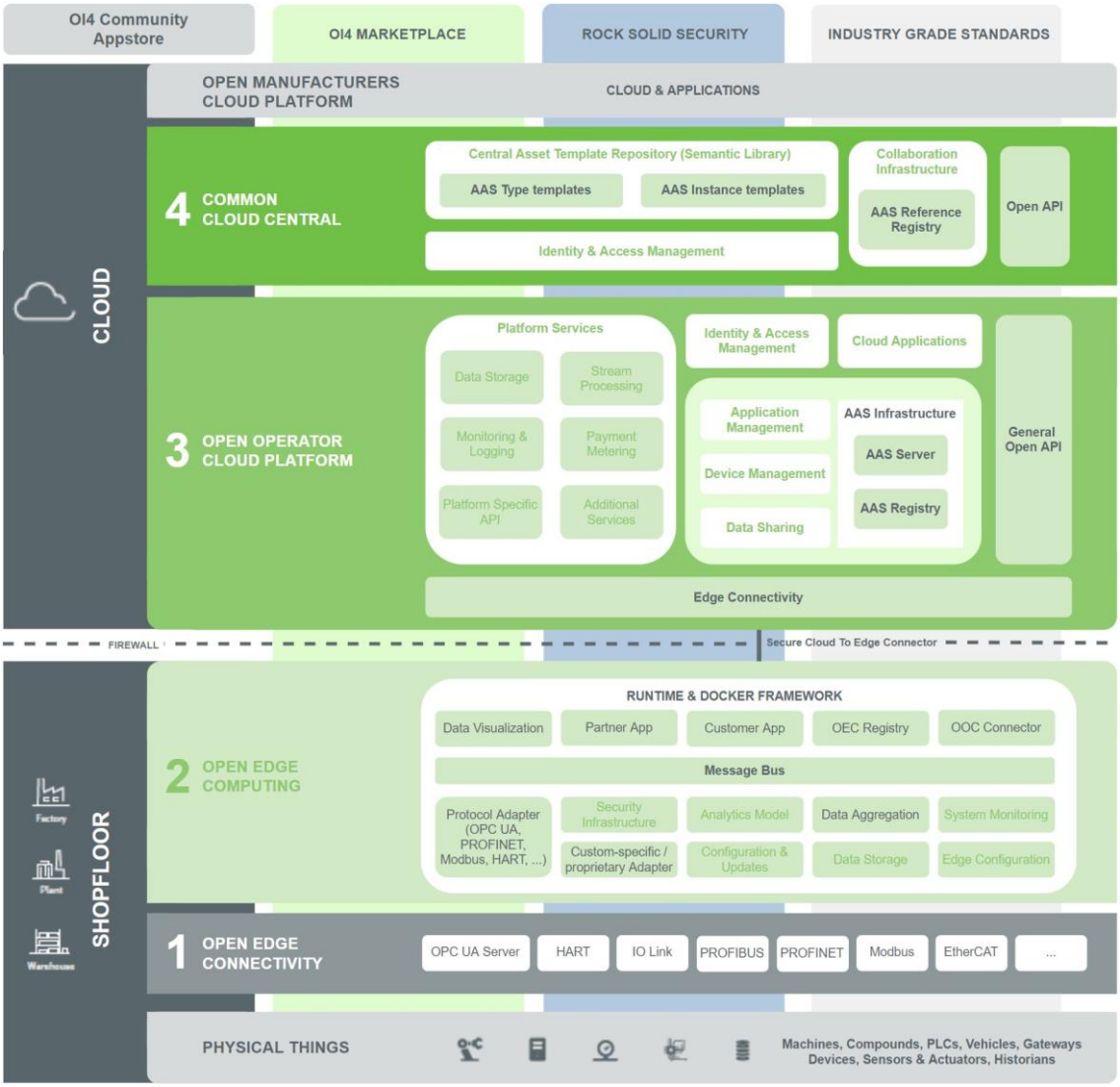

Figure 1 Open Industry 4.0 Alliance Reference Architecture

Aiming at providing a modular, flexible, and scalable architecture for an extensive set of scenarios, the Open Industry 4.0 Alliance concept defines all components of the Open Edge Computing layer to be containers. For the first implementations, the Docker containers has been chosen as container technology. Additionally, a Message Bus is defined as mandatory

component in the Open Edge Computing layer as it ensures a standardised communication and interaction between the different containers.

Following industry best practices on IoT and Edge Computing stacks, the Message Bus is based on the MQTT technology. MQTT technology allows high performance and robust data exchange even for lightweight computing resource availability. However, in order to address ambiguities in the topic definition for MQTT, the Open Industry 4.0 Alliance defines topic structures to ensure interoperability within Open Industry 4.0 Alliance compliant containers.

For the payload of the MQTT messages in the Message Bus, the Open Industry 4.0 Alliance technical committee has agreed on OPC UA payload formats. One reason for this technical decision is to establish future-proof Open Industry 4.0 Alliance solutions as OPC UA implementations on the shop floor will increase (green-field implementations for Industry 4.0 scenarios). Furthermore, OPC UA definitions are considered as best practice in the industry, especially regarding device information in OPC UA Part 100, and are adapted by other industrial groups as well.

While the mentioned technologies helps the Open Industry 4.0 Alliance's solutions developed according this guideline to fulfil interoperability requirements, a common approach for addressing and distinguishing components has to be provided as this goes beyond neutral technology definitions of the used standards. For this purpose, an `oi4Identifier` and a Master Asset Model are defined as mandatory components for every entity that interacts with the Message Bus. These are core concepts of Open Industry 4.0 Alliance and are described in the following chapters.

With regard to cyber security the IEC 62443 is considered looking at the overall automation system. Looking at the edge computer as such this is specialized considering DIN SPEC 27070.

2.1 Introduction to Edge Computing

Edge Computing refers to a paradigm for decentralized data processing that takes place on the edge of the network to support the Cloud during the operations of business-critical scenarios. From a manufacturing perspective, this means that Edge Computing is performed at the shop floor and ensures a continuous synchronisation with the cloud.

In both ways, an Edge computer (or Edge node) is a gateway from the Operational Technology (OT) used on Control level to the Information Technology (IT) used on Enterprise level - or specifically in the cloud. In this way, a so called Edge-Cloud Continuum is spanned. By using an infrastructure with both the Cloud and the Edge, applications and data can be distributed to the location where it fits best (e.g to ensure the continuity of business scenarios). An Edge computer typically has access to the field network and therefore is close to the place where data is produced. Compared to Cloud Computing this has several advantages:

- To save bandwidth, data may be processed at the edge instead of transmitting it to the cloud.
- To ensure continuity of business process even with intermitted connectivity.
- To address latency and limited bandwidth, Edge Computing allows reactions in real-time and fast descision making.
- For sensitive data Edge Computing provides a way to keep data in the premises.

Typically an edge computer is managed by the cloud. In most of the cases, applications running at the edge are also operated, managed and orchestrated from the cloud.

NOTE *it is important to differentiate edge computing or edge computers to traditional on premise systems. While edge computing or edge nodes requires a connection to cloud systems, traditional on premise systems or solutions do not necessarily are connected to the cloud.*

From a cybersecurity perspective, the standards IEC 62443 and DIN SPEC 27070 are to be observed in edge computing.

2.2 Introduction to MQTT

"*MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.*"

Quote from the official MQTT 3.1.1 specification.

2.2.1 Standardization

MQTT has become an OASIS standard in 2014 and an ISO/IEC standard in 2016 (ISO/IEC 20922:2016). MQTT v3.1.1 standard is stable and, in particular, provides mechanisms for a secured communication. MQTT is well supported through various libraries and implemented in several programming languages. Both Open Source as well as commercial offerings based on the MQTT open standard are available in the market.

The Open Industry 4.0 Alliance decided to use the MQTT standard in version 3.1.1 for their Open Edge Computing Message Bus.

2.2.2 Communication principles of MQTT - Pub/Sub Mechanism

The architecture of MQTT is based on the publish/subscribe pattern. The producers of messages (so-called publishers) and the recipients of messages (so-called subscribers) are decoupled by a MQTT broker. The MQTT Broker is responsible to deliver a message to the correct recipients and manages the individual connected clients. The broker is able to send a single message sent by a publisher to various subscribers (1:n). Both sending and receiving clients are permanently connected to the MQTT Broker via a TCP connection - that is how the broker can push a message to interested clients with minimum delay.

NOTE *These basic principles are compatible to using OPC UA over MQTT, especially when utilizing OPC UA PubSub, and using the OPC UA JSON encoding. This chapter focuses on MQTT, but the other technologies mentioned here are also utilized, as will become apparent in later chapters.*

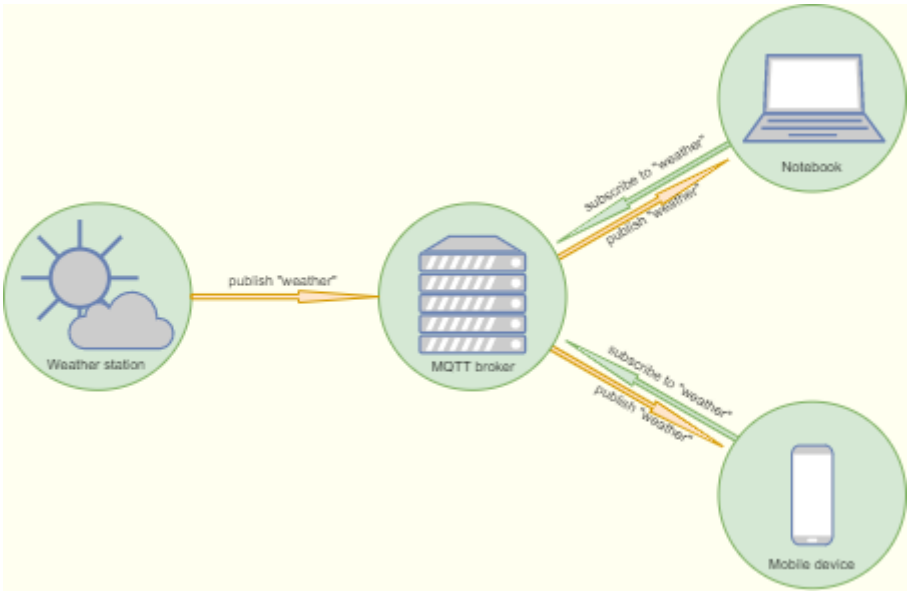

Figure 2 How the pub/sub mechanism works

2.3 Introduction to OPC UA PubSub

"PubSub enables the use of OPC UA directly over the Internet (Wide Area Networks) by utilizing popular data transports like MQTT and AMQP while retaining its key OPC UA end-to-end security and standardized data modeling advantages. Similarly, PubSub also enables use of the User Data Protocol (UDP) for establishing low-latency, loss tolerating connections on LANs."

Quote from the official site: https://opcconnect.opcfoundation.org/2018/04/opc-foundation-announces-opc-ua-pubsub-release/

OPC UA PubSub has been published in February 2018 as Part 14 of the OPC UA specification. As already described in the previous Section 2.2.2, *OPC UA PubSub* leverages the Publish/Subscribe communication pattern to achieve higher flexibility by decoupling sender and receiver with a message-orientated middleware (rf. https://github.com/OPCFoundation/UA-.NETStandard/blob/master/Docs/PubSub.md). The Ethernet-Standard IEEE 802 has been extended for real-time functionality. The OPC UA PubSub offers the real-time capabilities by using TSN (Time-Sensitive-Networking).

2.4 Introduction to containerization

Containerization refers to packaging of an application together with all its dependencies into a so-called container. A container is a standardized unit of software which allows to execute an application in any environment that provides the necessary components (e.g., runtime) for it. Typically, such environment offering a host system is a GNU/Linux based computer.

A container makes sure that an application is executed isolated from other containers and the host system. Isolation ensures that crash of a container is isolated from other containers at the same host and, consequently, avoids any harm to the host system or other containers.

To facilitate isolation, each container has a private file system which is provided by a so-called container Image. An image contains everything necessary to run an application. This includes the binaries (or code) together with all dependencies like libraries or runtime as well as any supporting files.

A container runs natively on Linux. It shares the Kernel with the host system and other containers. By special means of the Linux kernel the process running inside of a container is isolated whilst host resources are accessible. Thus, an application running in a container

doesn't require more memory than an application running directly on the host. This is the reason what makes containers lightweight, in particular if compared to virtualization e.g. using Hypervisors / Virtual Machines.

Open Industry 4.0 Alliance utilizes container technology to package its edge applications. The usage of containers is a de-facto industry standard in context of cloud as well as edge applications since containers technology provides the required security mechanisms. Another benefit is the portability of applications that containers make possible.

Open Edge Computing applications running as containers communicate with each other using MQTT. Any Open Edge Computing application is a container with an MQTT client as its public interface.

Further information introduction container technology can be found here https://www.docker.com/resources/what-container.

Alliance specific requirements are available in chapter 6 of this document.

2.5 Cyber Security

The IEC 62443 is a standard addresses cyber security topics of the automation system as whole, while DIN SPEC 27070 addresses specific needs of edge computing scenarios. Both standards are described in the following sections.

2.5.1 IEC 62443

The IEC 62443 is a family of standards which deals with the cyber security of Industrial Automation and Control Systems. It covers all necessary topics from protection of staff to technical configuration up to conformity with laws and regulations.

IEC 62443 has become a major standard for industrial cyber security. It contains all essential topics and is internationally recognized. Thus, it is the reference of various standards and norms for which serves as a basis.

An essential concept of IEC 62443 is the definition of four security levels. Each security level defines a set of requirements to bring security to one of the defined levels. The security levels build on one another.

Security Level	Description
1	Protection against casual or coincidental violation
2	Protection against intentional violation using simple means with low resources, generic skills and low motivation
3	Protection against intentional violation using sophisticated means with moderate resources, high skills and moderate motivation.
4	Protection against intentional violation using sophisticated means with extended resources high skills and high motivation.

Table 1 IEC 62443 - security level

At the time of this writing, the means described in this guideline are supposed to fulfill Security Level 1.

2.5.2 DIN SPEC 27070

DIN SPEC 27070 defines requirements and a reference architecture of a security gateway for the exchange of industry data and services.

In terms of the Open Industry 4.0 Alliance a security gateway is an edge computer which connects OT and IT.

The security relevant requirements if DIN SPEC 27070 are based on the requirements from IEC 62443. Thus, the more general requirements from IEC 62443 are applied for the more specific scope of application of a security gateway.

DIN SPEC 27070 defines three security profiles which include a mapping to four the Security Levels of IEC 62443.

Security Gateway Profile	Description
Basic	The intended use of “Basic“ profile is for scenarios with low demand of security, e.g. for demonstration or test purposes The “Basic” profile is in general mapped to Security Level 1 of IEC 62443.
Trusted	The intended use of “Trusted“ profile is for scenarios where the protection of the processed and transmitted data is essential. The “Trusted” profile is generally mapped to to Security Level 3 of IEC 62443.
Trusted Plus	The “Trusted Plus” profile adds protection against manipulation by administrators using technical means to the Trusted Profile.

Table 2 DIN SPEC 27070 - security gateway profile

At the time of this writing, the means described in this guideline are supposed to fulfill Security Profile “Basic“.

3 The oi4Identifier

A basic element of any Open Industry 4.0 Alliance system is the oi4Identifier that every asset and every piece of software shall contain and which allows the unique identification of an asset across different systems. Unless the manufacturer defines the ProductInstanceUri the same as the oi4Identifier, it can be generated at any time of the assets lifecycle and provides a stable point of reference for that particular asset in all dealings with other Open Industry 4.0 Alliance systems throughout the asset's lifetime. In case of defining the ProductInstanceUri the same as the oi4Identifier, it is created at the beginning of the lifecycle

The oi4Identifier can be generated by using the content of the asset's nameplate. Therefore, it is possible to regenerate the identifier if necessary using the creation rules defined in 3.1. Generation of the described oi4Identifier allows the usage for brownfield and greenfield.

The oi4Identifier is compliant to DIN SPEC 91406 but might differ to other existing or upcoming DIN SPEC 91406 conform, vendor specific Ids, often called “ProductInstanceUri”.

NOTE *The oi4Identifier has no reference to the installation location, nor does it contain any other topological information.*

3.1 Structure of the oi4Identifier

The oi4Identifier in the form of `<ManufacturerUri/Model/ProductCode/SerialNumber>` represents a DIN SPEC 91406 compliant URI and can be broken down into its constituent parts, resulting in further filtering options by topic, such as all applications from manufacturer A or the assets of type B.

The structure of the oi4Identifier contains the following elements, which are separated in the identifier by a forward slash "/".

`ManufacturerUri`:

Type: URL encoded string which must necessarily be written in lowercase letters.

The `ManufacturerUri` is the part of the manufacturer URL were the scheme (e.g. <https://>) and the third level domain (e.g. <www.>) are omitted. Consequently, it must be of the form `<manufacturer>.<tld>`.

For a generated `ManufacturerUri` from any kind of vendor ID, such as done by a protocol adapter, the globally available `ManufacturerUri` table should be used as reference (see 3.2 for details).

NOTE *In contradiction to DIN SPEC 91406, the URL-encoded string must necessarily be written in lower case. Upper and lower case would trigger several problems because <example.com> would be different to <Example.COM>.*

NOTE *The* `ManufacturerUri` *shall refer to an existing internet domain that is under the management of the manufacturer. A manufacturer may use subdomains of its internet domain for use as* `ManufacturerUri` *if it deems it necessary. It is only required that the site is under control of the manufacturer and the provision of data is not required.*

`Model`:

Type: URL encoded string

Should provide `Model` information. If the Master Asset Model (see 4) of the identified asset contains the `Model` property at the time the oi4Identifier is generated, the value shall be used here.

`ProductCode`:

Type: URL encoded string

Contains a product code, often called order number, which allows the manufacturer to unambiguously identify the specific type information for an asset. If the Master Asset Model (see 4) of the identified asset contains the `ProductCode` property at the time the oi4Identifier is generated, the value shall be used here.

`SerialNumber`:

Type: URL encoded string

Containing the serial number of an asset. If the Master Asset Model (see 4) of the identified asset contains the property `SerialNumber` at the time the oi4Identifier is generated, the value shall be used here.

NOTE *If deploying a container, which by its nature has no serial number and can be instanciated multiple times, use the container name as* `SerialNumber` *as defined in 6.3.*

NOTE *If there are no values for a specific oi4Identifier element, that element shall be represented by "nd" (no data).*

NOTE *If assets are used without a unique serial number and "nd" is used instead, only one instance of this asset can be reliably tracked. This problem has to be solved on application level.*

NOTE *According to DIN SPEC 91406, the path of an URL, which starts right after the* `ManufacturerUri`*, is case-sensitive. Therefore the three elements* `Model`*,* `ProductCode` *and* `SerialNumber` *of an oi4Identifier are case-sensitive.*

NOTE *The whole oi4Identifier, including the 3 forward slashes and the URL-encoded components* `ManufacturerUri`*,* `Model`*,* `ProductCode` *and* `SerialNumber` *must not exceed 255 characters.*

NOTE *Requirement 5.4 (forbid character combinations defined in another RFC for special functions) of the DIN SPEC 91406 cannot be fulfilled, because it is not possible to test against all existing and upcoming RFCs!*

3.2 Additional provisions regarding the oi4Identifier

The `oi4Identifier` as a lasting reference to a particular asset has to fulfill many roles in an Open Industry 4.0 Alliance system. To that end, any implementation should ensure that the `oi4Identifier` for a particular asset is either known to all involved systems or can be easily retrieved. The information given about manufacturer, type, order number and serial number should ensure unambiguous identification.

The `oi4Identifier` is a URI according to DIN SPEC 91406 and shall fulfill all requirements given there as well as the requirements described in this document. Special attention must be paid to restrictions regarding the `ManufacturerUri` and the usage of forward slashes.

Due to the fact that some manufacturers forego the concepts of `Model` and `ProductCode` and instead use a globally unique `SerialNumber` for each product they sell, it is extremely important to keep in mind that an element of the `oi4Identifier` can contain "nd". This serial number is then sufficient to identify and distinguish the product.

To enable local generation of the `oi4Identifier` in the Open Edge Computing layer of the Alliance architecture, some information has to be held by the adapter(s) of this layer. For example, the pattern to be used for generating an identifier for a manufacturer must be known to the generating instance. As mentioned above, some manufacturers do not need `Model` or `ProductCode` to properly identify assets. However, other manufacturers, have a critical need for this information. For this reason, a table will be provided in the future, which summarizes the manufacturers known to the Open Industry 4.0 Alliance and specifies the mandatory elements of the identifier associated with that particular manufacturer.

As another example, most real-time networks give the manufacturer ID as a number encoded by a technology-specific table. Since this table differs significantly for most real-time network technologies, a table has to be provided for the adapters which translates the manufacturer ID into the required format.

NOTE *Some manufacturer may have a so called ProductInstanceUri or other DIN SPEC 91406 related IDs. Even OPC UA Part 100, which defines a name plate, similar to Open Industry 4.0 Alliance's Master Asset Model (4), has such an ID. It is not allowed to overwrite this IDs with the oi4Identifier because different IDs might be used in different context.*

4 The Master Asset Model

To provide further standard information for the assets in an Open Industry 4.0 Alliance system, the Master Asset Model has been defined. It provides a joint reference to relevant asset information that goes beyond the limited identification option provided by the oi4Identifier and is therefore useful, for example, for storing and restoring version information.

NOTE *The properties that can be acquired through the Master Asset Model are equivalent to the properties defined in OPC UA Part 100 as part of the optional VendorNameplate Interface (Part 100-4.5.2). If a vendor has implemented the optional VendorNameplate Interface from OPC UA Part 100, they shall use the same data content for implementing the Master Asset Model information.*

For the sake of completeness, the properties of the Master Asset Model are described as follows:

`Manufacturer`:

Type: LocalizedText
Access: read only
Requirement: Mandatory

The name of the manufacturer of an asset. It should be noted that uniformity in the name representation should be a goal, as typically there are different ways of writing a company name regarding all legal appendices and so forth. In order to avoid shortcomings in identification due to this, the `ManufacturerUri` is also given.

`ManufacturerUri`:

Type: String
Access: read only
Requirement: Mandatory

This property shall provide a valid URI to RFC 3986. It shall adhere with all rules for an `oi4Identifier` as far as the `ManufacturerUri` is concerned (3.1). These rules are:
1. `ManufacturerUri` shall be a domain name.
2. `ManufacturerUri` may contain subdomains.
3. The `ManufacturerUri` given for this property shall also be used for the `oi4Identifier` of the asset if generated from an Alliance compliant application.

`Model`:

Type: LocalizedText
Access: read only
Requirement: Mandatory

The property `Model` shall provide the name of the product that the Master Asset information belongs to. Note that model names often are not sufficient to unambiguously identify a product. In order to avoid shortcomings in identification due to this, the `ProductCode` is also given. The value given for this property shall also be used for the `oi4Identifier` of the asset.

`ProductCode`:

Type: String
Access: read only
Requirement: Mandatory

A `ProductCode` has the purpose to unambiguously identify the model of an asset. Typically, every manufacturer has their own schemes for the generation of product codes. The `ProductCode` given for this property shall also be used for the `oi4Identifier` of the asset.

`HardwareRevision`:

Type: String
Access: read only
Requirement: Mandatory

The `HardwareRevision` shall be given according to the manufacturer-internal revision number of the hardware. If for a physical asset a manufacturer does not have a formalized hardware revision, it shall be set by default to "undefined". For any subsequent hardware revisions, this value shall be incremented in an appropriate way. A recommended best practice is to use a dot-separated four numbers schema.

NOTE *An application (software only) might be listed as an asset and therefore also has a nameplate. In this case, the* `HardwareRevision` *shall be set to an empty string to detect the differences between hardware and software only assets!*

NOTE *A hardware with undefined or unknown hardware revision shall set* `HardwareRevision` *to* `undefined`*.*

`SoftwareRevision`:

Type: String
Access: read only
Requirement: Mandatory

The `SoftwareRevision` shall be given according to the manufacturer-internal revision number of the software installed on the asset. If for any asset a manufacturer does not have a formalized software revision, it shall be set by default to "undefined". For any subsequent software revisions, this value shall be incremented in an appropriate way. A recommended best practice is to use a dot-separated four numbers schema.

NOTE A pure mechanical part might not have any software component. In this case, the `SoftwareRevision` should be set to an empty string!

`DeviceRevision`:

Type: String
Access: read only
Requirement: Mandatory

The `DeviceRevision` is a conglomerate revision that manufacturers may use to distinguish certain shipment configurations of their devices. Typically, a device revision will contain a specific hardware revision and a specific software revision, where the latter may change during the lifecycle through updates.

`DeviceManual`:

Type: String
Access: read only
Requirement: Mandatory

A `DeviceManual` property shall contain a URL that points to a human-readable manual document for the asset. The format of the document is currently unspecified, but a best practice would be a PDF document at the time of the writing of this document.

`DeviceClass`:

Type: String
Access: read only
Requirement: Mandatory

This property is a placeholder for storing semantic references regarding the application domain role of the asset in question. Some roles will be defined by the Alliance in order to easily administer an Open Industry 4.0 Alliance system. Where no such predefined role exists, a semantic referencing framework can be employed to specify the purpose of the asset for the domain in question.

NOTE *The Alliance does not define the content of this field, since it belongs to the provider of the asset. However, best practice for Alliance-compliant applications is to name the serviceType (8.1.2) here in the form of OI4.<serviceType>.*

`SerialNumber`:

Type: String
Access: read only
Requirement: Mandatory
The serial number shall be a unique identifier for an instance of an asset. It is the manufacturer's responsibility to provide globally-unique serial numbers for all their products. If the assets do not have a unique serial number, it shall be generated in an appropriate way to be able to have instances of equal types. In case of generating a serial number, it is up to the underlying technology to generate a unique serial number, as there is no unique way to define it. Unique serial number can be generated by using unique hardware IDs such as MAC addresses or ports and have to be defined in context. The `SerialNumber` given for this property shall also be used for the `oi4Identifier` of the asset.

`ProductInstanceUri`:

Type: String
Access: read only
Requirement: Mandatory

The `ProductInstanceUri` shall contain a unique ID provided by the manufacturer. This ID, likely a DIN SPEC 91406 conform URL, might not be equal to the `oi4Identifier`.

NOTE *The* `ProductInstanceUri` *shall not be overwritten by the* `oi4Identifier` *specified in section 3.1.*

NOTE *According to the OPC UA specification the* `ProductInstanceUri` *must not exceed 255 characters.*

`RevisionCounter`:

Type: Int32
Access: read only
Requirement: Mandatory

Changes to the configuration of the master asset information shall lead to an increment of the revision counter by one.

`Description`:

Type: LocalizedText
Access: read only
Requirement: Optional

A short localized text as a description can be added here. This object is not defined in OPC UA Part 100, but might be useful for a short overview of the available assets on Open Edge Computing level.

NOTE *Each component, regardless if it is a physical thing or an application, which was detected by the Alliance framework, should get an* `oi4Identifier` *and a Master Asset Model. This is necessary, because each asset (hardware and software) generates costs and needs maintenance during its lifetime.*

5 Overall process description

This chapter describes the overall message exchange processes as they are to be used for the onboarding and sending of process data in an Open Industry 4.0 Alliance context.

Central elements for that are the existence of a Message Bus (see 7) and a so called OEC Registry (A1) on the Open Edge Computing platform. All other components are use case driven but interacting with these two components.

5.1 Layer interactions by use cases

An overview of the layers of the Alliance' architecture has been given in figure 1. The layers have to collaborate for most use cases. In order to give an understanding of these interactions, the following sections elaborate on major use cases for the interaction between layers.

5.1.1 Asset onboarding

Assets need to be connected on the Open Edge Connectivity layer in order to be automatically detectable. For most technologies, however, after being plugged in, an asset still has to be actively detected, because assets do not actively send notifications that they are connected. Therefore, the Open Edge Computing layer has to actively scan its network and receive some form of Asset identifier. On the Open Edge Computing layer, an `oi4Identifier` (3) and a Master Asset Model (4) have to be generated. This process is described in more detail in section 5.2.1.

As a result of the internal generation processes in the Open Edge Computing layer, an onboarding message is sent to the Open Operator Cloud Platform. Depending of the implementation of the Asset Administration Shell, the onboarding sequence may look like shown in figure 3. This sequence diagram shows an overview of the default flow, there may be additional/alternative mechanisms and applications.

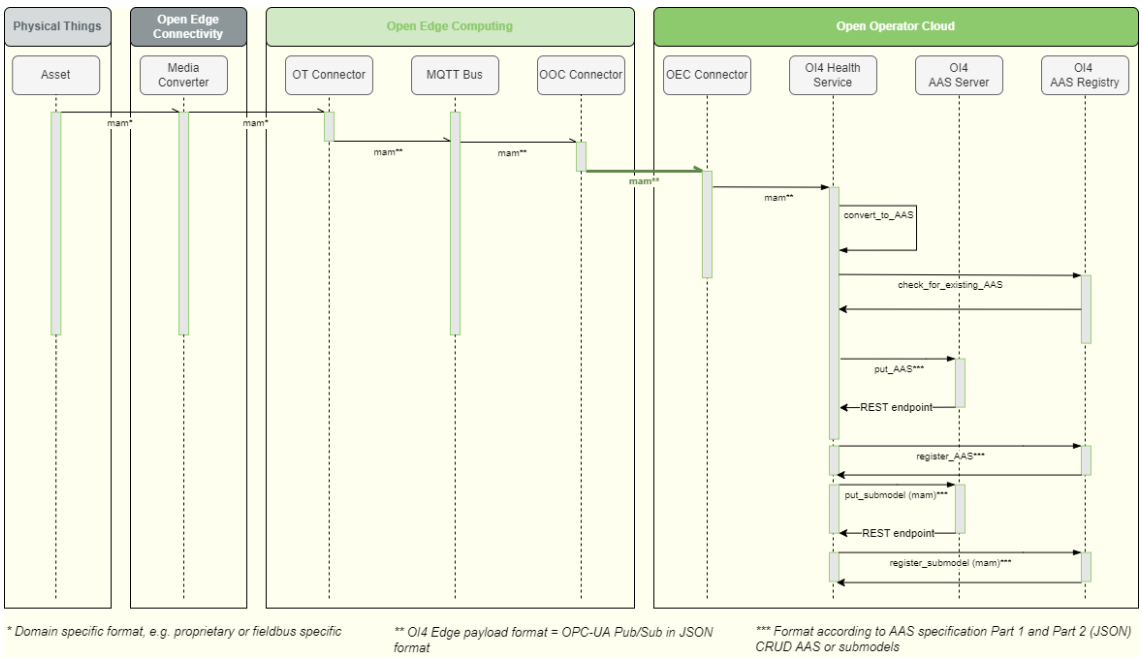

Figure 3 Sequence diagram for asset onboarding

5.1.2 Condition monitoring (health)

Assets need to be connected on the Open Edge Connectivity layer in order to be automatically detectable. For most technologies, however, after being plugged in, an asset still has to be actively detected, because assets do not actively send notifications that they are connected. Therefore, the Open Edge Computing layer has to actively scan its network and receive some form of health information. On the Open Edge Computing layer, an Alliance compliant health information have to be generated. This process is described in more detail in section 5.2.2.

As a result of the internal processes in the Open Edge Computing layer, a health message is sent to the Open Operator Cloud Platform. The OOC Platform creates and updates a sub model “health” which is related to the digital twin of the asset instance.

The process steps for condition monitoring are illustrated in the sequence diagram in figure 4. This sequence diagram shows an overview of the default flow, there may be additional/alternative mechanisms and applications.

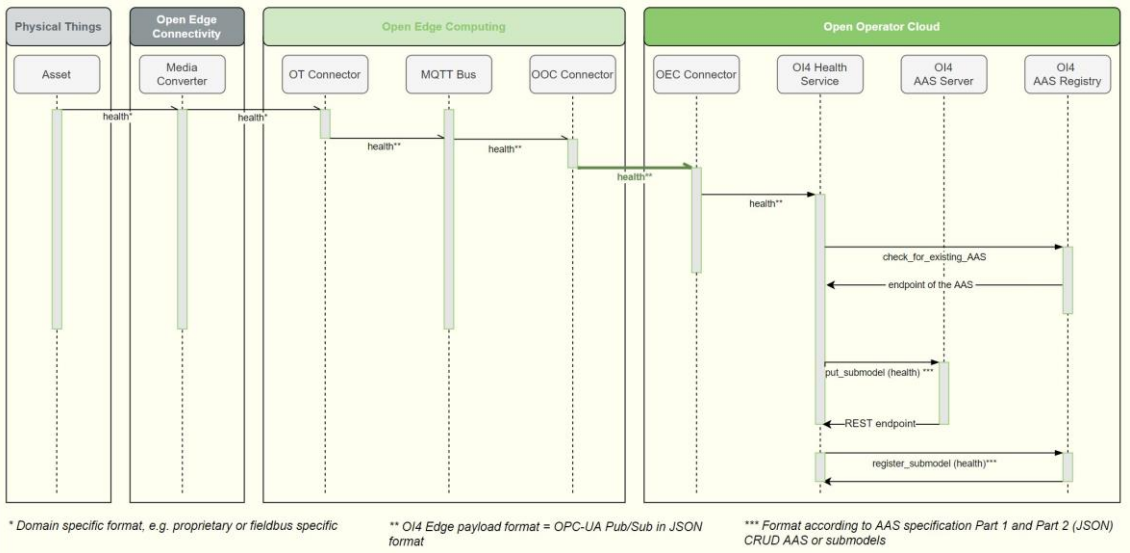

Figure 4 Sequence diagram for condition monitoring (health)

5.1.3 Handling process data

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The handling of process data in the Alliance architecture can be done in very different ways. Particularly, data may, but does not have to, be stored, aggregated or connected in a time series. The Technical Committee of the Alliance has devised sequence diagrams for three scenarios that can and will likely happen around the processing of real-time data that is received from the Open Edge Connectivity layer. This section will describe each of them.

Please note that the Open Edge Computing layer has undergone further discussion since these sequence diagrams were devised. The diagrams assume that an event is sent from a shop floor asset. This might also be just a regular process data interval and will not necessarily take the form of an event. Likewise, the component labeled “Device Driver” in the diagrams has since been relabeled as “Protocol Adapter”, fulfilling more functions than originally envisioned. The diagrams will be updated accordingly in later versions of this document.

An explanation of the basic data ingestion in the Open Edge Computing layer with mapping to message type is given in section 5.2.3.

The first use case shown is just plain process data processing. Data that is being ingested in the Open Edge Computing layer is forwarded to the Open Operator Cloud platform. From there, it is made available to other platforms in a standardized OI4 format.

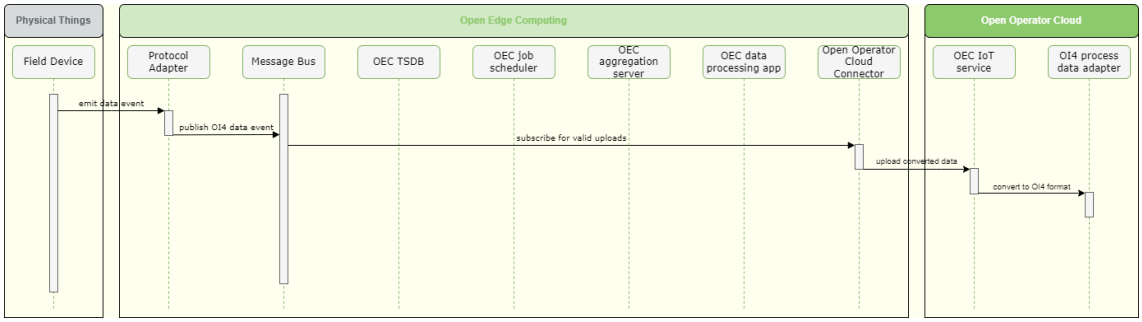

Figure 5 Plain Process Data sequence diagram

The second use case repeats the steps from the first one but adds the task of edge data processing. Data processing apps in the Open Edge Computing environment can receive the ingested data, process it according to their configuration and publish the result for upload to the Open Operator Cloud platform via the Message Bus.

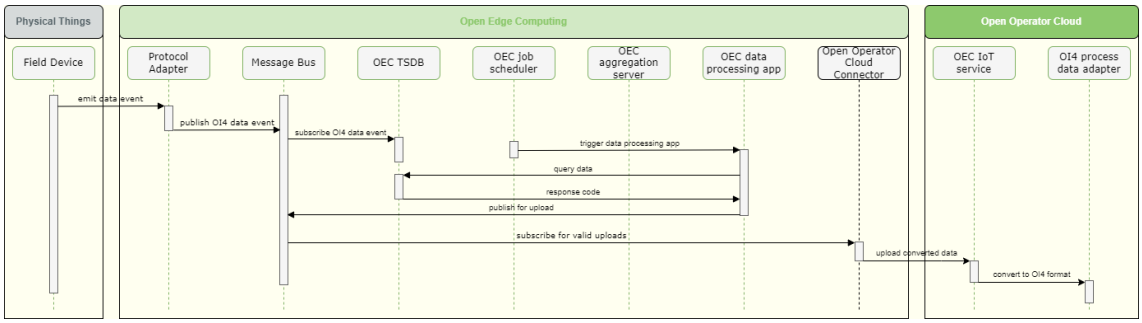

Figure 6 Edge Data Processing sequence diagram

The third use case extends the data ingestion process with a Store-and-Forward process. Data that is ingested is also stored in a time series database. An aggregator service can then request several data items from the time series database and aggregate them through various means. The result can, again, be published to the Open Operator Cloud platform via the Message Bus.

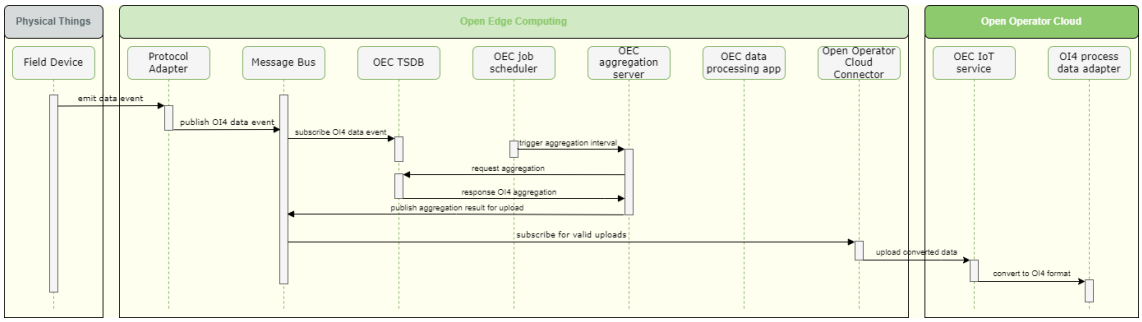

Figure 7 Store and Forward sequence diagram

5.2 Detailed container interactions inside the Open Edge Computing layer

The Open Edge Computing layer is constituted of containers that perform the various tasks relating to the OI4 use cases. These containers interact via a Message Bus. The Message Bus is to be implemented as an MQTT Broker and many details about this implementation are given

in chapter 2.2 and 7. What follows is a detailed description of the two use cases Onboarding of a Field Device and Process Data Ingestion as they are currently defined.

5.2.1 Onboarding of a field device

When a new device is connected to an operational level communication interface, it does not always cause an event. In order to adapt to the technological differences between Open Edge Connectivity solutions, the various adapters (protocol, proprietary, or device-specific adapters) are responsible for detecting a newly connected device. This can be done via event detection when possible or through active means of detection (e.g. polling).

Next, the protocol adapter has to generate the `oi4Identifier` (3) and the Master Asset Model (“mam”, see chapter 4) of the new asset. Therefore it evaluates the provided data from the new device against a reference list with vendor IDs, etc.

Once, the Master Asset Model is created, the protocol adapter sends a "mam" message (10.1.1) to the Message Bus. This message can be used by all containers listening to "mam" messages to recognize the newly connected device.

The Open Operator Cloud Connector receives the "mam" message or asks the OEC Registry for a list of available “mams”. It recognizes, that this particular “mam” has not yet been sent to the Open Operator Cloud and sends it out with an onboarding notice according to the AAS service.

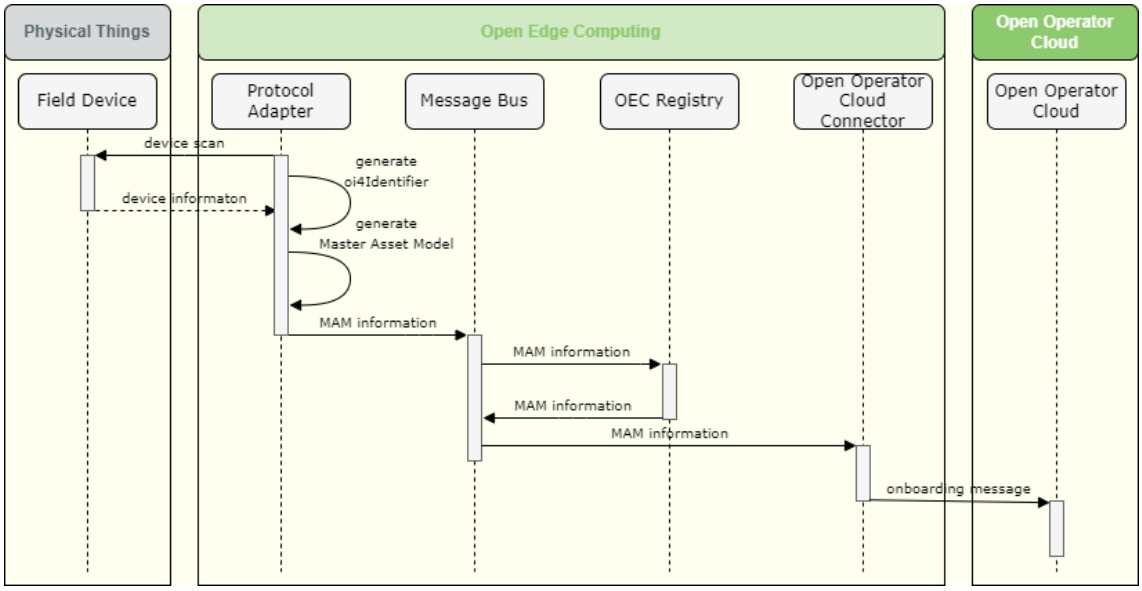

Figure 8 Sequence diagram for the onboarding of a field device

5.2.2 Condition monitoring (health) of a field device

When a device is connected to an operational level communication interface, it does not always cause an event. In order to adapt to the technological differences between Open Edge Connectivity solutions, the various adapters (protocol, proprietary, or device-specific adapters) are responsible for detecting connected devices and its health state. This can be done via event detection when possible or through active means of detection (e.g. polling).

Once, the health information is available, got changed or it is time to send a heartbeat, the protocol adapter sends a "health" message (10.1.2) to the Message Bus. This message can be used by all containers listening to "health" messages to monitore the connected device.

The Open Operator Cloud Connector receives the "health" message or asks the OEC Registry for a list of available “health”. It recognizes, that this particular “health” needs an update or not and sent it to the Open Operator Cloud according to the AAS service.

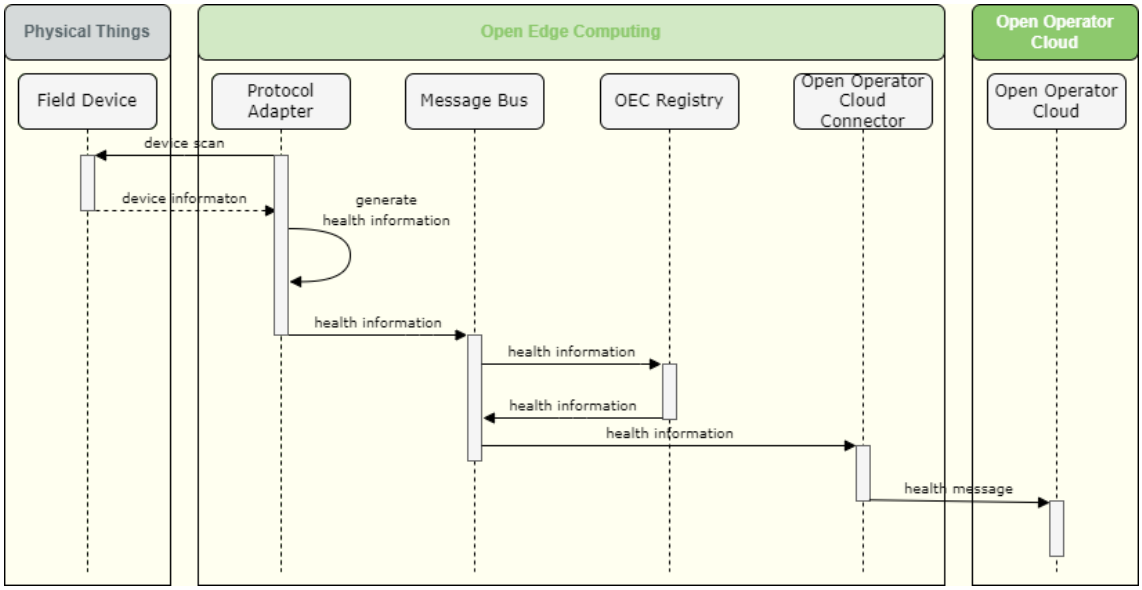

Figure 9 Sequence diagram for the condition monitoring (health) of a field device

5.2.3 Process data ingestion of a field device

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

Process data is typically ingested in regular intervals in order to generate time series that can be evaluated. However, for on-demand measurements, a triggered ingestion is also possible. The handling of both regular and triggered ingestions is the task of the protocol, the proprietary or the device-specific adapter. For typical real-time networks, it is possible to acquire data cyclically but this has to be pursued actively by the OEC layer.

The Protocol Adapter publishes the ingested process data as a "data" object (10.1.7) on the Message Bus. The "data" object contains in its topic all source information for the process data.

The Open Operator Cloud Connector receives the new "data" object and evaluates its configuration regarding any demands for sending it to the Open Operator Cloud.

Any containers who subscribed to the specific tag of the "data" object receive the ingested process data and process it according to their function.

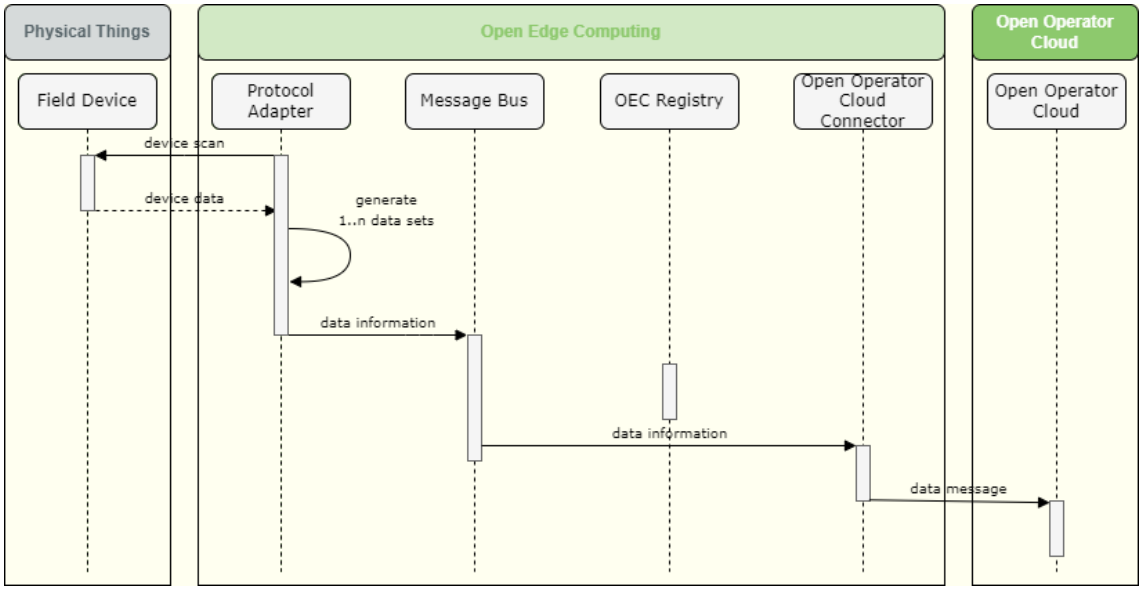

Figure 10 Sequence diagram for the process data ingestion of a field device

6 Container environment

To enable all Open Industry 4.0 Alliance members to interoperate on Open Edge Computing platform with each other, the Alliance has to define the expectation from the container point of view.

This chapter will explain the common settings and mechanisms for security, data exchange, and configuration of container related things.

NOTE *The guideline does not prescribe a particular container runtime environment, it just specifies the required settings. In case of concrete examples or snippets, the chapter references the docker reference documentation (https://docs.docker.com/reference/). However, this is just an example.*

NOTE *Be aware, settings might be available only internally, but not in a distributed system or settings such as ip address, hostname, paths, stores might change during operation.*

6.1 Container storage

Container volumes are designed to have persistent storage that is independent of a container. The difference between a container volume and data inside of a container is, that if you delete a container or start a new container from an updated image, the data on the container volume is still available.

NOTE *A container volume is basically a folder or file on the file system of the container host. Therefore, it is hard to protect the folder and its content - at least the host system is able to manipulate or even delete folder and/or content.*

A container volume can be assigned to a container during its startup sequence. One and the same container volume can be assigned to several containers at the same time or to only one container.

6.1.1 Message Bus storage

The Message Bus storage shall be used to make the public certificate and the settings of the MQTT broker available to the container and their MQTT clients. The Guideline does not specify how this is done on the host. It just specifies how it is mapped to the container to clearly specify the expectation from the container point of view.

Target path	Description
/etc/oi4/mqtt	Folder which contains the mqtt settings and the public certificate of the broker **ADVISORY** *Do not store any private keys here!*

Table 3 Message Bus storage path

The Message Bus storage shall contain at least the following files:

File	Description
broker.pem	Public certificate of the broker.
broker_ca.pem	Local CA used to sign the MQTT broker certificate. NOTE *If no local CA is present, the global CA (see 6.1.2) is used.*
broker.json	JSON coded file containing the settings of the MQTT broker.

Table 4 Message Bus storage content

The broker.json contains the following elements:

`address`:

Value range: `<ip addredd or hostname>`
Type: String
Requirement: Mandatory

Defines the IP address or hostname of used MQTT broker.

`secure_port`:

Value range: `<UINT16>`
Type: UINT16
Requirement: Mandatory

Defines port, where the MQTT broker is listening to.

`max_packet_size`:

Value range: `<INT32>`
Type: INT32
Requirement: Mandatory

Defines the maximum size of a MQTT payload in kiB. A NetworkMessage, containing several DataSetMessages shall not exceed this size.

NOTE *A value of 256 kiB (= 262,144 bytes) seems to be practical for most mid-range Open Edge Computing platforms.*

6.1.2 OI4 certificate storage

In case the system provides a Certificate Authority (CA) for signing certificates, the CA should be made available by storing it as follows.

Target Path	Description
/etc/oi4/certs	Storage for certificates. Such as client certificates for client certificate based authentication and CA certificate to validate other certificates. ADVISORY *Do not store any private keys here!*

Table 5 OI4 CA storage path

The Message Bus storage shall contain at least the following files:

File	Description
ca.pem	CA used for validation of certificates.
<containername>.pem	Client certificate of a specific application should be named as described in 6.2.

Table 6 OI4 CA storage content

6.1.3 Secret storage

Container runtime environments provide special mechanisms to handle secrets (e.g. passwords). In particular docker swarm or Kubernetes provide a feature for imprinting secrets into a container. The guideline does not prescribe these mechanisms - it just specifies how this is seen from the container's perspective.

In an Open Industry 4.0 Alliance compliant application the MQTT credentials shall be made available to the container under the following path:

Target file	Description
/run/secrets/mqtt_credentials	If present, it contains base64 encoded username and password, which is used for MQTT authentication. In the format: `base64(<username>:<password>)`
/run/secrets/mqtt_private_key.pem	If present, it contains the private PEM encoded key to the according client certificate in the OI4 certificate store.
/run/secrets/mqtt_passphrase	If present, it contains the base64 encoded passphrase for the private key.

Table 7 Secret storage

An example of how this can be accomplished with a docker-compose file is shown below:

```
secrets:
 - mqtt_credentials

secrets:
  mqtt_credentials:
    file: ./my_secret_credentials_on_host_filesystem.txt
```

6.1.4 Application specific storages

Application specific container storages for persistent data and/or configuration may be necessary for some applications.

In case the system provides a persistent layer, the application should use the following folders to make use of it.

Target Path	Description
/etc/oi4/app	Application specific configuration shall be stored here
/opt/oi4/app	Application specific data shall be stored here

Table 8 OI4 specific application storage path

Storages might use some kind of mechanism, such as the Docker mount type `volume` for persistent or mount type `tmpfs` for non-persistent content.

RECOMMENDATION *Attention must be paid to access permissions: To guarantee the data ownership, only one container shall have access to its own application specific volume. Because container volumes are not protected against external read/write access in general, the application has to take measures to detect data manipulation and might protect it (e.g. encryption) if necessary.*

6.2 Container name conventions

Each Alliance compliant container shall have a well-defined name to identify it. Because the container name is system wide unique, it can be used to address different use cases such as being reused as MQTT clientId within the system.

An additional need for Open Industry 4.0 Alliance compliant containers is to have a unique oi4Identifier (see 3), which makes it necessary to have a serial number or a equivalent for that.

To combine these needs, the container name must be unique worldwide according to the definition of the Alliance, but must also not be longer than 63 characters. Best practice is a name schema like <DefaultContainerName>_<uuid>.

The container name is given during creation of the container and shall be made available to the container by setting the hostname of the container.

The container name is used for several things:

- hostname of the application
- MQTT clientId (7.2.3)
- The name of the MQTT client certificate (6.1.2)
- As serial number inside of the oi4Identifier (3) and appId (8.1.3)

NOTE *It is likely possible, that users which have more than one Open Edge Computing platform to manage, are reusing same names for same containers (e.g. OEC Registry container might be named the same in each system). With this, all instances of a specific application are ending up with using the same virtual serial number and the same MQTT clientId. Application instances with identical oi4Identifier as a result of using the same virtual serial number, are not able to onboard differently - therefore no unique digital twin instance is possible. In worst case several instances are represented through the same and single digital twin.*

NOTE *To avoid several instances of a container with the same container name, it is best practice to use a UUID as postfix on top of a self defined container name such as <DefaultContainerName>_<uuid>.*

NOTE *The rfc1123 rules for assigning hostname must also be followed when assigning container names for DNS resolution to work.*

An example of how the container name and hostname can be set in Docker is shown below:

```
docker run --name <DefaultContainerName>_<uuid> --hostname
<DefaultContainerName>_<uuid> hello-world:latest
```

6.3 Container Image Integrity

It is common practice to pull container images based on tags. Tags are intentionally changeable, a good example is the `latest` tag that usually points to the latest released version of an image. The tag stays the same, but the actual image, the tag is pointing to, may change over time. Furthermore, it is also often used to deliver security updates for base images, even for images with semantic versioning.
On the downside, the actual downloaded image may be different from the intended one, if it has changed in between. Therefore, the installed images can differ among individual edge devices, even if the same tag was used.
Nevertheless, tags provide a good way to select a specific version of an image and prevent human errors. Therefore the usage of tags within the Alliance is permitted and the default.

RECOMMENDATION *In cases where a dedicated, immutable version of an image is needed, digests can be used. A digest is the unique identifier of an image as a SHA256 hash.*

6.4 Container networks

Container deployment and management are not possible without building a reliable network. There are four main network types in docker:

- Closed Network / none Network
- Host Network
- Bridge Network
- Overlay Network

In the following, the different types are briefly described and their advantages and disadvantages are listed. An overview of the network environments already running, using a Docker container environment as an example can be called with the command `docker network ls`.

```
PS C:\User> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
fbfa578081ad   bridge    bridge    local
10b78e0e3c3a   host      host      local
37c2dffba462   none      null      local
PS C:\User>
```

Closed Network / none Network

This network type is used when the container does not need to connect to a network. The container does not get a network interface built in and works isolated from all other containers on our host. This is then called a closed container. There is only a loopback interface for the container and no interface hanging on a network.

Advantage	Disadvantage
It provides the highest level of network protection, but is only suitable if the container does not require network access.	This is not a good choice if a network or internet connection is required, which is usually the case in an Alliance compliant container.

Table 9 Advantage/disadvantage – Closed network

Host Network

With this network type, the container is given the network properties of the host.

Advantage	Disadvantage
Containers running in the host network stack should have higher performance than those going through the docker0 bridge and iptables port mapping.	It is the least protected network model, adds the container to the host network stack.
Low level Ethernet communication is possible.	Containers deployed on the host stack have full access to the host interface.

Table 10 Advantage/disadvantage – Host network

Bridge Network

The Bridge Network is Docker's default network. This network allows containers to communicate with each other and the outside world.

Advantage	Disadvantage
Connection to the Internet	OT connectors might have restrictions using low level Ethernet communication.
Connection of containers with each other	

Table 11 Advantage/disadvantage – Bridge network

Overlay Network

The Overlay Network is used to implement network configurations between multiple hosts. All other network types can only be implemented on a single host.

To implement an Overlay Network, the Container must be running in swarm mode and there must be a key-value store available.

Since building such a network requires in-depth knowledge of more advanced infrastructure systems for container environments, it will not be discussed further here.

Recommendation

In host networking, the host sends all communication messages over named pipes. However, this method can pose an increased security risk, as all traffic passes through the same containers without any separation mechanisms.

The second variant, the bridge network, avoids this by creating an internal network that connects to the external one.

NOTE *For the use of containers in an Alliance compliant edge computing environment, the use of bridge network types should be prefered.*

7 General requirements on MQTT

The Open Edge Computing framework uses MQTT as Message Bus. This Message Bus is used as standard communication channel between Alliance compliant containerized applications.

A huge range of MQTT brokers and clients are available for Open Edge Computing. Most of them will be able to fulfill the needs of the Alliance.

The following describes the framework conditions and default behavior for an MQTT client that is to be used in the Alliance' context.

Because of the different needs of Open Edge Computing providers, the MQTT broker can be implemented both as a host component and as a container.

7.1 Protocol version

Open Industry 4.0 Alliance requires the use of MQTT v3.1.1.

The Alliance defined MQTT v5.0, which was not published at the time of the decision, is not yet widespread enough and full featured libraries are rarely available. This decision is observed by the workgroups and technical committee of the Open Industry 4.0 Alliance for further technology steps.

The chosen MQTT standard gives the Alliance Members the opportunity to use the following key features:

- Edge indicated communication - no open ports, beside MQTT (s. 7.2.1)
- TLS with CA certification - for secure communication (s. 7.2.2)
- Authentication via username and password (s. 7.2.2)
- Optional MQTT broker access control list

Additionally, the Alliance is taking care about an

- OT centric topic namespace definition, which are service oriented
- OT centric payload definition, using OPC UA pub/sub format in JSON

7.2 Client connection setup

Each client, connected to a MQTT broker, should follow some basic rules for setting up the connection.

7.2.1 Broker address and ports

In the context of Open Edge Computing, the MQTT broker runs likely on the same computing platform as the clients connected to it. In the Alliance’s overall technical architecture, it constitutes an implementation of the Message Bus component.

Therefore all clients connect to the broker which is reachable via:

IP address:

The IP address or hostname of the MQTT broker is system-dependent and must be made available to any container by using the Message Bus storage (6.1.1).

Port(s):

As default setting, the port 8883 for secure communication is used. Over the Message Bus storage the default might be set to a system-dependent value (6.1.1).
Non-secure connections over port 1883 is not supported.

All data transmissions between assets and the Open Edge Computing layer (and possibly some going beyond it if security can be provided) have to use the MQTT broker.

7.2.2 Security settings

Broker side security settings

The MQTT connection to the broker is always encrypted. TLS in version 1.2 or higher is mandatory. The Broker preferable shall use a certificate which is signed by a well-known or user specific Certificate Authority.

NOTE The Broker certificate is provided via Message Bus storage as explained in 6.1.1.

NOTE The CA certificate is available via OI4 certificate storage as explained in 6.1.2.

Client side security settings

The Client must authenticate against the Message Bus either by username and password or with client certificate (mTLS).

NOTE The secrets are provided via a secret storage. Details are provided in 6.1.3.

NOTE The client certificate is provided via the certificate storage as described in 6.1.2.

In case a privat key for an existing client certificate and username/password are available, the client certificate based authentication is used.

NOTE *The Username must not contain reserved characters according to rfc3986, such as colon* `:`, *pound* `#` and others.

7.2.3 Client identifier

Each client has the possibility to define a unique ID to communicate with the broker. So far, this is not relevant on a technical level and in many cases, some random ID gets generated from the client SDK or it may even be left empty.

However, each Open Industry 4.0 Alliance conform application has a so-called `oi4Identifier` (3), which is unique and which could be used as a client identifier for the MQTT client. But to avoid problems with length restrictions in some rare MQTT implementations, the name of the container should be used, which is unique, present and accessible in any Alliance compliant container (6.2).

7.2.4 Keep alive

The MQTT protocol has a built-in keep alive mechanism to monitor the underlying TCP connection. This function is used among other things to send the will message (7.3.5) if necessary.
The keep alive interval, counted in seconds, is configurable and should be set up to a default value of 60 seconds.

7.2.5 Clean session (as opposed to persistent session)

The Clean Session flag (boolean) represents a feature for the subscriber. It switches on or off the Persistent Session functionality.

A Persistent Session exists over the lifetime of a TCP connection (e.g. mobile network is temporarily interrupted). All missed QoS1 and QoS2 messages (7.3.1) are sent to a subscriber when the connection is re-established. QoS0 messages are lost. To use this feature, a client identifier (7.2.3) must be set.

In the context of Open Industry 4.0 Alliance, it is highly recommended to set the Clean Session flag to TRUE by default.

Here are some reasons why:

- A TCP connection to a locally installed environment should always be excellent.
- A queuing mechanism could force performance and stability issues on small and mid-range Open Edge Computing platforms.
- It has a much bigger impact on implementation on application side.

7.3 Client message setup

The MQTT client is controlled by an application that defines all messages to publish or to subscribe to.

There are different message types, such as Birth, Will and application-driven messages. Every single message has settings and might have special behavior as described in the following subsections.

7.3.1 Quality of Service level

The Quality of Service (QoS) level is a definition between publisher and subscriber of a message that defines the guarantee of delivery for a specific message.

There are three QoS levels available:

- 0: „At most once" or „fire and forget"
- 1: „At least once"
- 2: „Exactly once" or „once and only once".

The publishing client defines the QoS level between client and broker. The subscribing client defines it between broker and client. It is important to remember it is not an end-to-end connection between the clients - therefore different QoS levels can be used.

In the context of Open Edge Computing, with always a very good connection quality over local networks, it shall be enough to use QoS 0 as a default level.

7.3.2 Retained messages

The Retained Message flag (boolean) represents a feature for the publisher. If the Retained Message flag is set, the last published message is kept at the broker, so that a client that subscribes later than the time the message was originally published still gets the "old" message - and does not have to wait for an update.

In context of the Alliance, the Retained Message functionality is switched off by default and shall not be used.

NOTE *Missed messages can be requested from clients through the get method defined in chapter 8.1.4. The Retained Message mechanism is not used for brokers because the get*

method on clients fulfills the requirements of the Open Industry 4.0 Alliance in a better way by using less resources.

7.3.3 Birth message

This is a message that will be published on the configured topic whenever the connection between client and broker is established.

In an Open Industry 4.0 Alliance context, the Birth message contains a NetworkMessage with a DataSetMessage of type Master Asset Model (4) of the application. It refers to the topic resource mam (8.1.5) and shares its payload.

Payload of DataSetMessage (9.2.3) of type mam for the birth message:

```
"Payload": {
  "Manufacturer": {
    "locale": "en-US",
    "text": "<manufacturer name>"
  },
  "ManufacturerUri": "",
  "Model": {
    "locale": "en-US",
    "text": "<model name>"
  },
  "ProductCode": "<product code>",
  "HardwareRevision": "<hardware revision>",
  "SoftwareRevision": "<software revision>",
  "DeviceRevision": "<device revision>",
  "DeviceManual": "<link to device manual>",
  "DeviceClass": "<device class>",
  "SerialNumber": "<serial number>",
  "ProductInstanceUri": "<product instance uri / possibly oi4Identifier>",
  "RevisionCounter": <revision counter>,
  "Description": {
    "locale": "en-US",
     "text": "<short description>"
  }
}
```

NOTE *The Birth message, which has the format of a mam message, should not contain the optional keys for Timestamp and SequenceNumber, because the content of this keys can not be pre-set.*

NOTE *See example in appendix B1.1.1.*

7.3.4 Close message

There is no definition in the MQTT specification for a message published before a graceful connection close happens.

An application-driven message should be published in OI4 context, before the connection is closed gracefully. Over this feature, the application can de-register gracefully (as opposed to Last Will) from the Open Edge Computing platform.

The close message contains the health information for a graceful disconnect of the application (9.3.2). It refers to topic resource health (8.1.5) and shares its payload.

Payload of DataSetMessage (9.2.3) of type health for the close message:

```
"Payload": {
  "health": "NORMAL_0",
  "healthScore": 0
}
```

NOTE *See example in appendix B1.1.2.*

7.3.5 Will message

The Last Will and Testament feature is used to notify other clients about an ungracefully disconnected client.

The broker stores the message until it detects that the client has disconnected ungracefully (timed out).

If the client disconnects gracefully by closing the connection, the broker discards the stored Will message.

The Will message contains the predefined health information for an ungraceful disconnect of the application (9.3.2). It refers to topic resource health (8.1.5) and shares its payload.

Payload of DataSetMessage (9.2.3) of type health for the will message:

```
"Payload": {
  "health": "FAILURE_1",
  "healthScore": 0
}
```

NOTE *The Last Will and Testament message, which has the format of a health message, should not contain the optional keys for Timestamp and SequenceNumber, because the content of this keys can not be pre-set.*

NOTE *See example in appendix B1.1.2.*

7.4 General behavior of MQTT on changes

Even though this is clear to any MQTT user, it should be mentioned again: Any change on a payload needs to be published without any external trigger on its corresponding topic. E.g. when a data tag gets updated or a health status changes, the related topic needs to be published immediately.

NOTE *Other behavior might be needed in some cases, e.g. floating analog values are rippling all the time but this has no effect for the process. Therefore different publication rules are applicable, which can be configured through publicationList (10.1.11).*

8 Topics definition

An MQTT topic consists of one or more **topic levels**, separated by a forward slash "/" – the **topic level separator**.

The whole topic is a UTF-8 eccoded string.

Be aware:

1. Each topic must contain at least one character.
2. White space is allowed, but should not be used.
3. The topic string is case sensitive.
4. The forward slash "/" is a reserved character for separation of topic levels.
5. A leading forward slash "/" introduces an unnecessary topic level with a zero character - therefore, never start a topic with "/".

The broker uses the topic to filter messages for each connected client. To enable subscription to multiple topics, the client is allowed to use wildcards in the topic he subscribes to.

Single-level wildcard: "+":

A single-level wildcard replaces one topic level.

Several "+", at least separated with separators, can be used.

Multi-level wildcard: "#":

The multi-level wildcard replaces many topic levels.

Only a single "#" at the end of a topic is accepted.

8.1 Topic namespace elements

For better interoperability of independently working applications from different members of the Alliance, it is a key position to use not only the same protocols but also the same semantics if not defined by the protocol itself.

With the exception of a few special characters, MQTT allows a largely free definition of topics. In the context of the desired interoperability, it is therefore essential to define a reliable schema/namespace to make the available data and functions generally accessible.

All applications, working according to the Open Industry 4.0 Alliance guideline, use the following schema:

```
oi4/<serviceType>/<appId>/<method>/<resource>/<subResource>/<filter>
```

NOTE *As described in the following sections, the elements* `subResource` *and* `filter` *might not be relevant for every use case. More specifically, their interpretation depends on the used* `resource`*.*

In the following sections all elements in the namespace are described in detail.

8.1.1 namespace element

All Open Industry 4.0 Alliance conform messages start with the namespace element `oi4`. This is the first mechanism to separate Open Industry 4.0 Alliance traffic from other traffic on the Message Bus.

NOTE *All topics starting with the namespace element shall be Alliance conform. Any other MQTT traffic shall not use this namespace.*

8.1.2 serviceType element

Each application that publishes or subscribes data represents a specific type of service. The `serviceType` element is used to roughly sort the applications.

NOTE *For the methods get, set and del, the* `serviceType` *of the destination, which has to react to these methods, has to be chosen. The* `serviceType` *of the publisher of the data itself must be chosen for the pub method.*

The following `seviceTypes` are currently addressed:

`Registry`:

Provides basic services, such as which applications and devices are available. See appendix A1 for details to the OEC Registry.

`OTConnector`:

Communicates with a subordinate network or individual assets from it.
An `OTConnector` likely makes assets available over the Message Bus, which can be listed from `Registry` and onboarded from `OOCConnector`.
A device which can communicate directly to the Message Bus represents the `serviceType` `OTConnector`.

`Utility`:

Provides utilities such as converters, configuration or other upcoming utility-like of services.

`Persistence`:

Provides mechanisms to persist data. It is not defined how the persistence will work - this `serviceType` just clarify what this application is made for, not how it is implemented.

`Aggregation`:

Aggregates subscribed data from different sources on the Message Bus and publishes them in a appropriated way. The sources are likely other applications which are providing raw data, which are getting aggregated to enriched data.

`OOCConnector`:

Communicates with the Message Bus on one side and with the higher-level Open Operator Cloud on the other side.

`ITConnector`:

Communicates with the Message Bus on one side and with an IT network or individual services on the other side.
An `ITConnector` in opposite to an `OTConnector` does usually not add assets such as physical things. An `ITConnector` usually connects software components such as MES/ERP systems.

NOTE *Applications, which fulfill several aspects of the* `serviceType` *classification might be available (e.g. an* `OTConnector` *might persist some of the master data of detected devices). The used* `serviceType` *should represent the main task of the application.*

8.1.3 appId element

The `appId`, which is constituted of the `oi4Identifier` (see 3) of the application, is used to uniquely identify the source of the information. The information source is the application that publishes the data to the Message Bus.

NOTE *For the methods get, set and del, the* `appId` *of the destination, which has to react to these methods, has to be chosen. The* `appId` *of the publisher of the data itself must be chosen for the pub method.*

NOTE *The* `appId`/`oi4Identifier` *is a structured identifier (3). Therefore it contains four pieces of information, separated with a forward slash. This makes it possible to use parts of the identifier to subscribe to it (e.g. subscribe to everything from a specific vendor).*

8.1.4 method element

Since the Open Industry 4.0 Alliance acts in a "REST-affine" environment, it makes sense to use the common terms "methods" and "resources" in the topic definition.

The method defines what is to be done. For example, a read, a write, an update or a delete is initiated.

`pub`:

Is used to publish a `resource` such as `data`, `mam`, `event`, ...

`get`:

Is used for the dedicated request of a `resource` such as `data`, `mam`, …
For example, to request asynchronously when the original publication was missed.

`set`:

Is used to write to a `resource` such as `data`, `metadata` or `config`.
A `set` should cause an `event` such as `notice` if successful and `warning` or `error` (application-specific) on fail.

`del`:

Used to delete `data`, `metadata`, `config` or `mam`.
A `del` should cause an `event` such as `notice` if successful and `warning` or `error` (application-specific) on fail.

All methods above are meant for pub/get/set/del standard resources (8.1.5.1) which are provided by applications or devices. Furthermore, Open Industry 4.0 Alliance provides method calls via a call/reply pattern over the Message Bus (8.1.5.2 and 8.1.5.3). Therefore the methods `call` and `reply` are needed.

NOTE *The technical workgroup discussed other technical solutions just as REST, but defined, that the actual function set should be available over a single technology stack => MQTT.*

`call`:

It is used to call a method via a call/reply pattern.
The method parameters are listed in the payload.

`reply`:

It is used to reply to the former method call.
The method results are listed in the payload.

8.1.5 resource element

The resource element specifies what a method element (8.1.4) is to be applied to. To differ between different kinds of resources Open Industry 4.0 Alliance distinguishes between resources and services.

A resource corresponds to a well defined partial model (8.1.5.1).

A service is mainly an OPC UA client/server like method calls, which implements a call/reply pattern over MQTT (8.1.5.2 and 8.1.5.3).

NOTE *The technical workgroup discussed other technical solutions just as REST, but defined, that the actual function set should be available over a single technology stack => MQTT.*

8.1.5.1 Resources

Each resource listed here represents a well-defined submodel that can be fully schema validated. This enables a simple and less error-prone exchange of information between any communication partners.

Depending on the resource, some submodels can only be published, others can also be requested or even written or deleted.

`mam`:

Used for request or publish applications or devices, represented via Master Asset Model (4, 9.3.1 and 10.1.1).

`health`:

Used to request or publish the health information of an application container or device (9.3.2 and 10.1.2).

NOTE `health` *will be published once during the application gets initialized, when the value of* `health` *changes, and every 60 seconds as a kind of heartbeat.*

`config`:

Used to maintain the configuration of an application container or device (9.3.3 and 10.1.3).

`license`:

Used to request or publish the license information of an application (9.3.4 and 10.1.4). Docker containers provide enough isolation to protect the Open Industry 4.0 Alliance framework from the risk that copyleft regulations extend across different containers.

`licenseText`:

Used to request or publish explicit license agreement text for a defined license (9.3.5 and 10.1.5).

`rtLicense`:

Used to request or publish the runtime licenses of an application container or device (9.3.6 and 10.1.6).

`data`:

Used to maintain data of an application container or device (9.3.7 and 10.1.7).

`metadata`:

Used to maintain metadata for the associated data (9.3.8 and 10.1.8).

`event`:

Used to publish events (see 9.3.9 and 10.1.9).
In many cases, this resource represents information, which might be available also via syslog and might be used for audit-trailing. In this case the category "syslog" (9.3.9.2) is used, which must be mentioned in the topic via `<subResource>`. The `<filter>` in the topic contains an eventFilter - in case “syslog” is used as `<subResource>`, `debug`, `informational`, `notice`, `warning`, `error`, `critical`, `alert` and `emergency` are possible as `<filter>`.

NOTE *Other* `<subResource>`*, beside “syslog” are available as event categories (9.3.9).*

`profile`:

Used to request or publish a list of all resources, which are supported at the application container or device (9.3.10 and 10.1.10).

`publicationList`:

Used to maintain a list of objects with available publications (such as data objects, mam objects, health objects, …) and their configuration (active/inactive, configurable, refresh interval) (9.3.11 and 10.1.11).

`subscriptionList`:

Used to maintain a list of objects with available subscriptions and their configuration (active/inactive, configurable) (9.3.12 and 10.1.12).

`interfaces`:

Used to request or publish a list of interfaces a physical device has (communication, power, buttons, LEDs, …) (9.3.13 and 10.1.13).

`referenceDesignation`:

Used to maintain the reference designation system of an asset (9.3.14 and 10.1.14).

8.1.5.2 Common services

Common services are domain-independent. That means each of this services is having a fully defined request structure (`methodsToCall`) and a fully defined response structure (`results`).

Common services are:

`fileUpload`:

Load a file (e.g. config, firmware, certificate, …) to an asset (10.2.1).

`fileDownload`:

Get a file (e.g. config, firmware, certificate, …) from an asset (10.2.2).

`firmwareUpdate`:

Initiate a firmware update (10.2.3).
How to perform the firmware update and when it is domain-specific and handled through the application, which offers the service.

`blink`:

Let a specific device blink to identify visually if the service is implemented (10.2.4).

`newDataSetWriterId`:

Requests a new, so far unused, DataSetWriterId from the Publisher, which consumes this method call (10.2.5).

Each service specifies what kind of method call will follow in the payload. For well-defined services, the service name is equal to the name of the method which is called.

NOTE *Advanced message bus users might use the name of the service for security reasons (e.g. who is allowed to subscribe to which topics).*

8.1.5.3 Specific services

Extended services are domain-dependent. That means each of this services is having a well-defined name, but domain-specific request structure (`methodsToCall`) and response structure (`results`).

Extended methods are:

`read`:

Read data or parameter from an asset (10.3.1).

`write`:

Write data or parameter from an asset (10.3.2).

`subscribe`:

Subscribe to specific information from an asset (10.3.3).
The information will be sent cyclically or on change (not yet defined, but a subscribed information should be published via `../pub/data/<tag>` to make it public available).

`unsubscribe`:

Unsubscribe from formerly subscribed information (10.3.4).

`genericMethod`:

The service genericMethod (10.3.5) is a place holder to implement any kind of services (remember, method name and parameters are set in the payload).

NOTE *Advanced message bus settings might suppress the usage of this topic to avoid usage of unknown functionality.*

Each service specifies what kind of method call will follow in the payload. For extended services, the service name might differ from the name of the method which is called.

ATTENTION: *This could be a security leak. The service read should be used to call methods like <readRegister>, <readParameter> or similar - but a compromised asset could use it for methods like <deleteAll>.*

NOTE *Advanced message bus users might use the name of the service for security reasons (e.g. who is allowed to subscribe to which topics).*

8.1.6 subResource element

The `subResource` element is an additional, more detailed information to the regular `resource`. Sub-resources are context-specific and define distinct message payloads and filters.

The Alliance defines two different scenarios.

Beside event, the `subResource` is an oi4Identifier (see 3), referencing an asset, providing the `resource`-related information.

The `subResource` for the event `resource` contains an event category. The payload of each category follows the same base schema, but differs in details and is interpreted category specific (9.3.9 ff.).

The mapping of the event category to the `subResource` topic key is according to the following list:

Event category	Sub-resource topic key
CAT_SYSLOG_0: Event contains a syslog specific message (9.3.9.2)	syslog
CAT_STATUS_3: Event contains OI4 Status Code (9.3.9.1)	status
CAT_NE107_2: Event contains a NAMUR NE107 signal (9.3.9.3)	ne107
CAT_GENERIC_99: Event contains user defined content (9.3.9.4)	generic

Table 12 Event categories

8.1.7 filter element

The `filter` follows a resource specific classification. It can be used to reduce the information to publish.

NOTE *The following resources do not make use of filter:* `mam`, `health`, `profile`, `rtLicense`, `interfaces` *and* `referenceDesignation`.

Resource: publicationList and subscriptionList

For the resources `publicationList` and `subscriptionList`, the filter can be used to name a resource type and following define, which information are of interest.

`<resourceType[/<tag>]>`:

Value range: `<resourceType followed by optional separator and tag>`
Type: String

Depending on the resource to filter for, the `filter` can be `<resourceType>` or `<resourceType>/<tag>`.
`<resourceType>` filters are relevant for resources without additional filter definitions (e.g. mam, health and others). `<resourceType>/<tag>` filters are relevant for resources, which do have filter definitions such as data, config and others.

Resource: data and metadata

For the resources `data` and `metadata`, the filter contains the name of the desired data set or metadata set.

`<tag>`:

Value range: `<url encoded String>`
Type: String

The `tag` filter can be used to filter messages for a particular data set or metadata set.

NOTE *The* `tag` *for a data object can be freely defined by the asset or application (e.g. derived from a device description). The Alliance defined a dataSet called* `oi4_pv` (see 9.3.7), *which can be used in addition to provide standardized assess to the process values.*

Resource: config

For the resources `config`, the filter contains the name of the desired data set.

`<context.name>`:

Value range: `<url encoded context.name>`
Type: String

The `context.name` from related config object contains the name of the DataSet and can be used to filter messages for a particular data set.

Resource: license and licenseText

For the resources `license` and `licenseText`, the filter contains the licenseId of the desired data set.

`<licenseId>`:

Value range: `<url encoded licenseId>`
Type: String

The `licenseId` filter can be used to filter messages for a particular data set.

Resource: event

For the resources `event`, the filter depends on the event category chosen in `subResource`. Each event category are having its own filters.

Resource: event, subResource: syslog

`<syslog_logLevel>`:

Value range: `<syslog Message Severities>`
Type: String

The `logLevel` filter can be used to filter messages for a particular logging level. The log levels are described by the RFC 3164 syslog Message Severities.
It is used to publish an syslog event message with a specific log level, like `informational` or `critical`.

`debug`:

According to RFC5424: Debug - debug-level messages
The `debug` level designates fine-grained informational events that are most useful to debug an application.

NOTE *This level shall not be used as default by any application, because logging over the Message Bus costs a lot of resources.*

`informational`:

According to RFC5424: Informational - informational messages
The `informational` level designates informational messages that highlight the progress of the application at coarse-grained level.

NOTE *Using* `informational` *for logging costs a lot of resources. Any application should use this feature carefully.*

`notice`:

According to RFC5424: Notice - normal but significant condition
The `notice` level should be used to reflect normal but significant events from outside the system, i.e. when a user interaction has taken place.

`warning`:

According to RFC5424: Warning - warning conditions
The `warning` level designates potentially harmful situations.

`error`:

According to RFC5424: Error- error conditions
The `error` level designates error events that might still allow the application to continue running.

`critical`:

According to RFC5424: Critical - critical conditions
The `critical` level should be used to reflect warnings from inside the system, i.e. an

unexpected state detected within the system.

`alert`:

According to RFC5424: Alert- action must be taken immediately
The `alert` level should be used to reflect serviceable internal system errors.

`emergency`:

According to RFC5424: Emergency - system is unusable
The `emergency` level designates very severe error events that will presumably lead the application to abort.

NOTE *It is best practice to make the log level configurable. One way to do so would be to add the publication of each log level to the publicationList (see 10.1.11) and disable the levels* `debug` *and* `informational` *by default. Another way would be to define the log level via docker environment variable (see 6.2).*

NOTE *The severity levels should be interpreted as follows:*
Internal Error *and* ***Warning****: The component has detected an error in it's hardware or software, which is not allowed during normal operation. If such an error occurs, the component must be exchanged or serviced. This kind of events map to the syslog severities „**Emergency**“, „**Alert**“ and „**Critical**“.*
External Error *and* ***Warning****: The component has detected a possibly problematic event coming from outside the component. A troubleshooting activity might be necessary and must be indicated to the user. This kind of events map to the syslog severities „**Error**“ and „**Warning**“.*
Event*: The component has detected a normal but important event during normal operation, which should be logged. In this category belong events like login, logout, configuration settings, startup notification and so on. This kind of event map to the syslog severities „**Notice**“ and „**Informational**“.*

Resource: event, subResource: status

`<statusCode>`:

Value range: `<OPC UA status code>`
Type: String

The `statusCode` filter can be used to filter messages for a particular status code. The classification is based on the leading part of the symbol name or the starting 0x00, 0x40 or 0x80 hex code of the OPC UA Status Code (see OPC UA Part 6-A.2).
It is used to publish an OPC UA status event message with a specific status code, like `good`, `uncertain` or `bad`.

Resource: event, subResource: ne107

`<NE107_statusSignal>`:

Value range: `<NAMUR NE107 status signal>`
Type: String

The `NE107_statusSignal` filter can be used to filter messages for a particular signal.
It is used to publish an NAMUR NE107 event message with a specific signal, like `normal`, `failure`, `checkFunction`, `outOfSpecification` or `maintenanceRequired`.

Resource: event, subResource: generic

`<generic_level>`:

Value range: `<generic level>`
Type: String

The `generic_level` filter can be used to filter messages for a particular level.
It is used to publish a generic event message with a specific level, like `high`, `medium` or `low`.

8.2 Minimum set of topic elements for Open Industry 4.0 Alliance compliance

Entities in an Open Industry 4.0 Alliance compliant system can be applications or devices. Both types of entities must be represented on the Message Bus with a minimum set of topic elements.

8.2.1 For applications

Any application, represented through its own MQTT client, must have a subset of functionality to be Open Industry 4.0 Alliance compliant.

The minimum subset of supported topic elements are:

- `namespace` (8.1.1)
- `serviceType` (8.1.2)
- `appId` (8.1.3)
- `method` (8.1.4)
- `resource` (8.1.5)
- `subResource` (8.1.6)
- `filter` (8.1.7)

The following methods (8.1.4) are mandatory for Open Industry 4.0 Alliance compliance:

- `pub`
- `get`

The following resources (8.1.5.1) are mandatory for Open Industry 4.0 Alliance compliance and must be listed in profile (9.3.10):

- `mam` (9.3.1)
- `health` (9.3.2)
- `license` (9.3.4)
- `licenseText` (9.3.5)
- `profile` (9.3.10)
- `publicationList` (9.3.11)

NOTE *Including the resource* `data` *to an application, makes support for* `metadata` *mandatory.*

NOTE *Including the methods* `set` or `del` *to an application, makes support for the resource* `event` *and category* `status` *mandatory.*

8.2.2 For devices

Any device must have a subset of functionality to be Open Industry 4.0 Alliance compliant.

In general, a device is represented through an application. Therefore some topic elements such as namespace, serviceType and appId are handled through this application and the device representation has no influence to it.

The minimum subset of supported topic elements are:

- `namespace` (8.1.1)
- `serviceType` (8.1.2)
- `appId` (8.1.3)
- `method` (8.1.4)
- `resource` (8.1.5)
- `subResource` (8.1.6)
- `filter` (8.1.7)

The following methods (8.1.4) are mandatory for Open Industry 4.0 Alliance compliance:

- `pub`
- `get`

The following resources (8.1.5.1) are mandatory for Open Industry 4.0 Alliance compliance and must be listed in profile (9.3.10):

- `mam` (9.3.1)
- `health` (9.3.2)
- `profile` (9.3.10)
- `referenceDesignation` (9.3.14)

NOTE *Including the resource* `data` *to a device representation, makes support for* `metadata` *mandatory.*

NOTE *Including the methods* `set` or `del` *to a device representation, makes support for the resource* `event` *and category* `status` *mandatory.*

9 Payload format

The Open Industry 4.0 Alliance agreed to use an OPC UA-conform JSON format for payload definition.

The JSON format of the OPC Foundation is described in Part 6 (Mappings).

OPC UA pub/sub is described in Part 14 (PubSub).

Part 3 (Address Space Model), Part 4 (Services) and Part 5 (Information Model) provide also relevant information.

NOTE *For a better interoperability of existing and future Alliance' compliant applications, the Open Industry 4.0 Alliance offers schema files in its official GitHub repository for all objects defined in this chapter. It is not allowed to extend one of these objects by user defined keys.*

Example message on the message bus:

- Message bus topic (8.1):
  `oi4/<serviceType>/<appId>/<method>/<resource>/<subResource>/<filter>`
- Message bus payload (9.1):

```
{
 "MessageId": "unixTimestampInMs-PublisherId",
 "MessageType": "ua-data",
 "PublisherId": "<serviceType>/<appId>",
 "DataSetClassId": "<GUID>",
 "correlationId": "<empty/omitted> or <initial MessageId>",
 "Messages": [
   {
     "DataSetWriterId": <UINT16>,
     "SequenceNumber": <UINT32>,
     "MetaDataVersion": {
       "majorVersion": <UINT32>,
       "minorVersion": <UINT32>
     },
     "Timestamp": "DateTime",
     "Status": <StatusCode>,
     "filter": "<filter>",
     "subResource": "<subResource>",
     "Payload": {
       <add your DataSet here>
     }
   }
 ]
}
```

Several examples of valid message bus topic and payload combinations are listed in chapter 10.

The following figure shows, how a OPC UA PubSub message is embedded into the MQTT transport protocol:

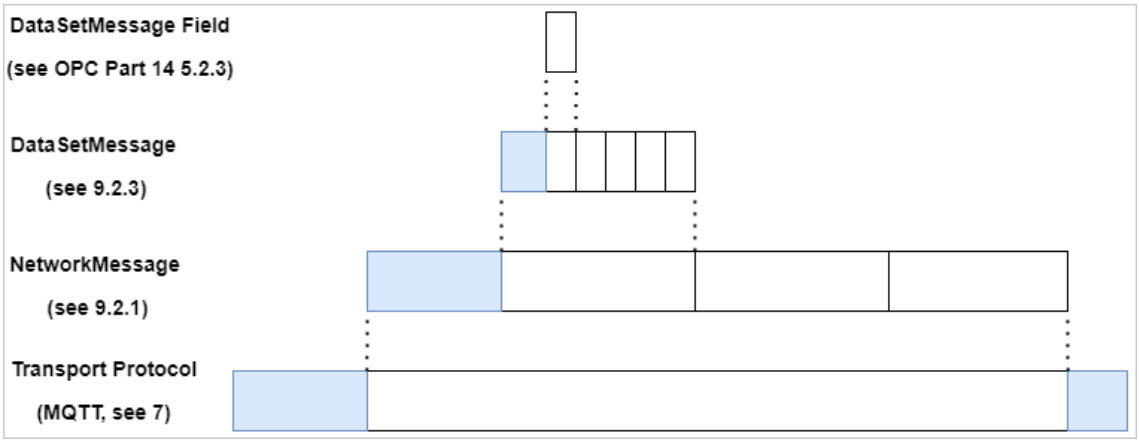

Figure 11 OPC UA PubSub message embedded in transport protocol

9.1 Structure of the defined MessageTypes

The `<MessageType>` field supports the following types: `ua-data` and `ua-metadata`. `ua-data` is small and serves for the transmission of recurring data. `ua-metadata` is extensive, since it completely describes the metadata of a data point.

In addition to the standardized `<MessageType>` types for OPC UA Part 14-7.2.3, Open Industry 4.0 Alliance defined the additional `<MessageType> MSG`. This type offers the possibility to implement a request-response pattern over the message bus. The `MSG` type adapts elements from the OPC UA client/server architecture to Open Industry 4.0 Alliance and its message bus.

9.1.1 Overview of the OPC UA conforming MessageType ua-data

The compact `ua-data` message consists of a number of elements and objects in accordance to the `DataSetMessage` type (9.2.3) and is used to transport a Alliance-defined or user-defined payload.

Non-exhaustive example of the structure of a `ua-data` message encoded in JSON:

```
{                                                                  NetworkMessage
                                                                   (see 9.2.1)
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "<GUID>",
  "correlationId ": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {                                                              DataSetMessage
                                                                   (see 9.2.3)
      "DataSetWriterId ": <UINT16>,
      "SequenceNumber": <UINT32>,
      "MetaDataVersion": {                                         ConfigurationVersionDataType
        "majorVersion": <UINT32>,                                  (see 9.2.5)
        "minorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Status": <UINT32>,
      "filter": "<filter>",
      "subResource": "<subResource>",
      "Payload": {
        <add your DataSet here>
      }
    }
  ]
}
```

Figure 12 A simple and valid ua-data JSON

9.1.2 Overview of the OPC UA conforming MessageType ua-metadata

The detailed `ua-metadata` message contains information about the structure and datatypes of a `ua-data` message. The structure is defined in accordance with the `DataSetMetaData` message defined in 9.2.2.

Non-exhaustive example of the structure of a `ua-metadata` message encoded in JSON:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",                    DataSetMetaData (see 9.2.2)
  "MessageType": "ua-metadata",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "<GUID>",
  "filter": "<filter>",
  "subResource": "<subResource>",
  "correlationId ": "<empty/omitted> or <initial MessageId>",
  "MetaData":
  {                                                                  DataSetMetaDataType (see 9.2.4)
    "name": "<String>",
    "description": {
      <localizedText>,                                               localizedText (see 9.2.7)
    },
    "dataSetClassId": "<GUID>",
    "configurationVersion": {
      <ConfigurationVersionDataType>,                                ConfigurationVersionDataType (see 9.2.5)
    },
    "fields": [
      <FieldMetaData>,                                               FieldMetaData (see 9.2.6)
    ],
    "namespaces": [
      "<namespace>",                                                 OPCF (see Part 14 A1.1)
    ],
    "structureDataTypes": [
      <StructureDescription>,                                        StructureDescription (see 9.2.9)
    ],
    "enumDataTypes": [
      <EnumDescription>,                                             EnumDescription (see 9.2.12)
    ],
    "simpleDataTypes": [
      <SimpleTypeDescription>,                                       SimpleTypeDescription (see 9.2.15)
    ]
  }
}
```

Figure 13 A simple and valid ua-metadata JSON

9.1.3 General MessageType MSG in accordance with OPC UA

The Alliance-specific message type `MSG` consists of a number of elements and objects, which are introduced here.

NOTE *Be aware, this message type is not conform to the OPC UA pub/sub specification, because OPC UA does not define or allow other* `<MessageTypes>` *types besides* `ua-data` *and* `ua-metadata` *for pub/sub. OPC UA client/server does define a* `<MessageType>` `MSG`*, but it is not used for pub/sub and contains many keys, which are not relevant for the pub/sub use*

case. The Open Industry 4.0 Alliance adopted the `<MessageType> MSG` *to use it for call/reply pattern.*

Non-exhaustive example of the structure of a `MSG` message encoded in JSON:

```
{                                                                  ServiceNetworkMessage
   "MessageId": "<unixTimestampInMs-PublisherId>",                 (see 9.2.16)
   "MessageType": "MSG",
   "PublisherId": "<serviceType>/<appId>",
   "DataSetClassId ": "<GUID>",
   "correlationId ": "<empty/omitted> or <initial MessageId>",
   "Message": {
      "methodsToCall": [                        ServiceParametersRequest (see 9.2.17)
         {                                                         CallMethodRequest
            "methodId": "<methodName>",                          (see 9.2.18)
            "inputArguments": [
               {<this content is specific for each method>}
            ]
         }
      ]
   }
}
```

Figure 14 A simple and valid request MSG JSON

A sample response to a `MSG` encoded in JSON:

```
{                                                                  ServiceNetworkMessage
   "MessageId": "<unixTimestampInMs-PublisherId>",                 (see 9.2.16)
   "MessageType": "MSG",
   "PublisherId": "<serviceType>/<appId>",
   "DataSetClassId ": "<GUID>",
   "correlationId ": "<empty/omitted> or <initial MessageId>",
   "Message": {
      "results": [                           ServiceParametersResponse (see 9.2.19)
         {                                                         CallMethodResult
            "statusCode": "<StatusCode>",                        (see 9.2.20)
            "inputArgumentResults": ["<StatusCode>"],
            "outputArguments": [
               {<this content is specific for each method>}
            ]
         }
      ]
   }
}
```

Figure 15 A simple and valid response MSG JSON

9.2 OPC UA objects, defined by OPC Foundation

This chapter explains the structure and data types of the objects used by the Open Industry 4.0 Alliance for the pub/sub communication over the Message Bus.

The following information is a subset of the OPC UA specifications Part 3, Part 4, Part 6 and Part 14. For further information it is highly recommended to relate to the linked OPC UA specifications.

9.2.1 NetworkMessage

The MQTT payload for `data` is an object of type NetworkMessage and is explained in OPC UA Part 14-7.2.3.2.

The NetworkMessage object contains the following elements (see also OPC UA Part 14-7.2.3.2-Table 91):

`MessageId`:

Value range: `<unixTimestampInMs-PublisherId>`
Type: String
Requirement: Mandatory

Example: `"1567062381000-OTConnector/company.com/model/productCode/4711"`

NOTE *The* `MessageId` *must be unique. The Open Industry 4.0 Alliance defiens the type as a combination of the current timestamp in ms precision with the* `PublisherId`*. In some rare cases, the system timestamp might not be precise enough to avoid sending packages with an unique* `MessageId`*. In these cases, the application must guarantee the uniqueness by providing an additional parameter (example.* `MessageId` = `<unixTimestampInMs><counter>-<PublisherId>`*).*

`MessageType`:

Value range: `"ua-data"`
Type: String
Requirement: Mandatory

Only `ua-data` is valid here.

`PublisherId`:

Value range: `<serviceType>/<appId>`
Type: String consisting of `serviceType` (see 8.1.2) and `oi4Identifier` (see 3.1) - separated by a `/`.
Requirement: Mandatory in Open Industry 4.0 Alliance context, but not in OPC UA context.

`DataSetClassId`:

Value range: `<GUID>` in accordance to OPC UA Part 6-5.1.3: 16 Byte as JSON string with separator (Part 6-5.4.2.7).
Type: String
Requirement: Optional

Example: `"f1875b4a-3209-431b-a38d-2df5758f92c8"`
The `dataSetClassId` allows to refer to the DataSetClass describing the structure of the message. The `dataSetClassId` identifies a well defined `DataSet` specified by the Alliance or some other standards and refers to related metadata.

For some recurring use cases, such as `MasterAssetModel`, fixed GUIDs are specified from the Alliance and must be used (A2).

NOTE *The* `DataSetClassId` *shall be present for all resources defined by the Open Industry 4.0 Alliance, which are having a* `DataSetClassId`*. This guarantees the availability of a fully schema verifiable messaging.*

`correlationId`:

Value range: `<empty/omitted>` or `<MessageId>`
Type: String
Requirement: Conditional

Shows the flow between the causal event and its consequences. The `correlationId` does not belong to OPC UA `DataSetMessage` according to Part 14-7.2.3.3-Table 92.

NOTE *The* `correlationId` *is filled in by the first consumer with the* `MessageId` *of the original message and then passed on from service to service until the message is no longer processed.*

`Messages[]`:

Value range: `<array of DataSetMessage>`
Type: DataSetMessage object (see 9.2.3)
Requirement: Mandatory

9.2.2 DataSetMetaData

The MQTT payload for `metadata` is an object of type `DataSetMetaData` and is explained in OPC UA Part 14-7.2.3.4.2.

The `DataSetMetaData` object defines the following elements (see also Part 14-7.2.3.4.2-Table 93):

`MessageId`:

Value range: `<unixTimestampInMs-PublisherId>`
Type: String
Requirement: Mandatory

Example: `"1567062381000-http://company.com/type/order/4711"`
Must be unique for any single package of this `PublisherId`.

NOTE *The* `MessageId` *must be unique. The Open Industry 4.0 Alliance defines the type as a combination of the current timestamp in ms precision with the* `PublisherId`*. In some rare cases, the system timestamp might not be precise enough to avoid sending packages with an unique* `MessageId`*. In these cases, the application must guarantee the uniqueness by providing an additional parameter (example.* `MessageId` = `<unixTimestampInMs><counter>-<PublisherId>`*).*

`MessageType`:

Value range: `"ua-metadata"`
Type: String
Requirement: Mandatory

Only `ua-metadata` is valid here.

`PublisherId`:

Value range: `<serviceType>/<appId>`
Type: String consisting of `serviceType` (see 8.1.2) and `oi4Identifier` (see 3.1) - separated by a `/`.
Requirement Mandatory

`DataSetWriterId`:

Value range: `<UINT16>`
Type: UInt16
Requirement: Mandatory

An identifier for `DataSetWriter` which published the `DataSetMetaData`. It is unique within the scope of a Publisher. The related `DataSetMessage` (9.2.3) to this `DataSetMetaData` contains the same `DataSetWriterId`.

NOTE *The* `DataSetWriterId` *is not persistent and can change on every power cycle of a DSWID*

`filter`:

Value range: `<filter>`
Type: String
Requirement: Mandatory

The `filter` is mandatory, but does not belong to OPC UA `DataSetMetaData` according to Part 14-7.2.3.4.2-Table 93. In combination with the used resource in the topic, the `filter`, together with the `subResource`, contains the readable reference to the `DataSetWriterId` and is identical to the filter in the topic (8.1.7).

NOTE *The* `filter` *helps to combine the* `MetaData` *with the related source. In OPC UA context this is done via* `DataSetWriterId`*, but this is not very intuitive and might need additional actions to get missing information via* `publicationList` defined in *9.3.11*.

`subResource`:

Value range: `<subResource>`
Type: String
Requirement: Mandatory

The `subResource` is mandatory, but does not belong to OPC UA `DataSetMessage` according to Part 14-7.2.3.3-Table 92. In combination with the used resource in the topic, the `subResource`, together with the `filter`, contains the readable reference to the `DataSetWriterId` and is identical to the `subResource` in the topic (8.1.6) if present.

NOTE *The* `subResource` *helps to combine the* `MetaData` *with the related source. In OPC UA context this is done via* `DataSetWriterId`*, but this is not very intuitive and might need additional actions to get missing information via* `publicationList` defined in *9.3.11*.

`correlationId`:

Value range: `<empty/omitted>` or `<MessageId>`
Type: String
Requirement: Conditional

Shows the flow between the causal event and its consequences. The `correlationId` does not belong to OPC UA `DataSetMessage` according to Part 14-7.2.3.3-Table 92.

NOTE *The* `correlationId` *is filled in by the first consumer with the* `MessageId` *of the original message and then passed on from service to service until the message is no longer processed.*

`MetaData`:

Value range: `<DataSetMetaDataType>`
Type: DataSetMetaDataType object (9.2.4)
Requirement: Mandatory

9.2.3 DataSetMessage

The DataSetMessage object contains the following elements (see also OPC UA Part 14-7.2.3.3-Table 92):

`DataSetWriterId`:

Value range: `<UINT16>`
Type: UInt16
Requirement: Mandatory

An identifier for `DataSetWriter` which published the `DataSetMessage`. It is unique within the scope of a Publisher. The related `DataSetMetaData` (9.2.2) to this `DataSetMessage` contains the same `DataSetWriterId`.

`SequenceNumber`:

Value range: `<UINT32>`
Type: UInt32
Requirement: Optional

A strictly sequentially increasing sequence number assigned to the `DataSetMessage` by the `DataSetWriter`.

NOTE `SequenceNumber` *might be of interest for resources with changing content, such as data, metadata, config, … More static like resources such as mam, health might not benefit from it.*

`MetaDataVersion`:

Value range: `<ConfigurationVersionDataType>`
Type: ConfigurationVersionDataType (9.2.5)
Requirement: Optional

The `MetaDataVersion` corresponds with the `configurationVersion` of a `DataSetMetaData` message (9.2.5).

NOTE `MetaDataVersion` *might be of interest for resources with changing parameter sets, such as data. Resources with fixed metadata set do not benefit from it.*

`Timestamp`:

Value range: `<DateTime>`
Type: String
Requirement: Optional

Example: `"2019-06-26T13:16:00.000+01:00"`
Timestamp of type `DateTime` according to ISO 8601-1:2019 and OPC UA Part 6-5.4.2.6. serialized as String.
The time of the data acquisition is indicated! Milliseconds might be of interest.

NOTE `Timestamp` *might be of interest for resources with changing content, such as health, data, metadata, config. More static like resources such as mam might not benefit from it.*

`Status`:

Value range: `<StatusCode>`
Type: UInt32
Requirement: Optional and shall not be shown, when Status = 0 => OK

Status code to be used as defined in OPC UA Part 4-7.34.2-Table 177 and CSV-File. We assume status "good" is equal to 0.

NOTE *The Status is not mandatory and should not be send, when the status is OK. When the Status is unequal to OK, please use the status codes, provided from OPC Foundation.*

`filter`:

Value range: `<filter>`
Type: String
Requirement: Conditional

Depending on related use case, the `filter` might be mandatory or optional, but does not belong to OPC UA `DataSetMessage` according to Part 14-7.2.3.3-Table 92. In combination with the used resource in the topic, the `filter`, together with the `subResource`, contains the readable reference to the `DataSetWriterId` and is identical to the filter in the topic (8.1.7) if present.

NOTE *The* `filter` *helps to combine the data in the* `Payload` *with the related source. In OPC UA context this is done via* `DataSetWriterId`*, but this is not very intuitive and might need additional actions to get missing information via* `publicationList` *defined in 9.3.11.*

NOTE *The* `filter` *is not needed for* `DataSetMessage` *of type* `pagination` *and* `locale`*. Furthermore, several resources such as* `mam` *or* `health` *and others does not make use of filter in Message Bus topic and* `DataSetMessage`*.*

`subResource`:

Value range: `<subResource>`
Type: String
Requirement: Mandatory

The `subResource` is mandatory, but does not belong to OPC UA DataSetMessage according to Part 14-7.2.3.3-Table 92. In combination with the used resource in the topic, the `subResource`, together with the `filter`, contains the readable reference to the `DataSetWriterId` and is identical to the `subResource` in the topic (8.1.6) if present.

NOTE *The `subResource` helps to combine the data in the `Payload` with the related source. In OPC UA context this is done via `DataSetWriterId`, but this is not very intuitive and might need additional actions to get missing information via `publicationList` defined in 9.3.11.*

`Payload`:

Value range: `<Object>`
Type: Object
Requirement: Mandatory

This object contains the name-value pairs specified by the `PublishedDataSet`.

NOTE *In general, all built-in data types should be possible, but it seems to be problematic to use ExtensionObject, Variant, DataValue, DiagnosticInfo and in some cases NodeId.*

9.2.4 DataSetMetaDataType

The `DataSetMetaDataType` contains the following elements (see also OPC UA Part 14-6.2.2.1.2-Table 3):

`name`:

Value range: `<name of DataSet>`
Type: String
Requirement: Mandatory

`description`:

Value range: `<localizedText>`
Type: LocalizedText (see 9.2.7)
Requirement: Optional

`fields[]`:

Value range: `<array of FieldMetaData>`
Type: FieldMetaData object (see 9.2.6)
Requirement: Optional

`dataSetClassId`:

Value range: `<GUID>` in accordance to OPC UA Part 6-5.1.3: 16 Byte as JSON string with separator (Part 6-5.4.2.7).

Type: String
Requirement: Optional

Example: `"f1875b4a-3209-431b-a38d-2df5758f92c8"`
The `dataSetClassId` allows to refer to the DataSetClass describing the structure of the message. The `dataSetClassId` identifies a well defined `DataSet` specified by the Alliance or some other standards.
For some recurring use cases, such as `MasterAssetModel`, fixed GUIDs are specified from the Alliance and must be used (A2).

NOTE *The* `DataSetClassId` *must be present for all resources defined by the Open Industry 4.0 Alliance, which are having a* `DataSetClassId`*. This guarantees the availability of a fully schema verifiable messaging.*

`configurationVersion`:

Value range: `<ConfigurationVersionDataType>`
Type: ConfigurationVersionDataType object (see 9.2.5)
Requirement: Optional

`namespaces[]`:

Value range: `<array of namespaces names>`
Type: String
Requirement: Mandatory

For details see OPC UA specification Annex A: Part 14-A1.1

`structureDataTypes[]`:

Value range: `<array of StructureDescription>`
Type: StructureDescription object (see 9.2.9)
Requirement: Mandatory

For details see OPC UA specification Annex A: Part 14-A1.3

`enumDataTypes[]`:

Value range: `<array of EnumDescription>`
Type: EnumDecription object (see 9.2.12)
Requirement: Mandatory

For details see OPC UA specification Annex A: Part 14-A1.4

`simpleDataTypes[]`:

Value range: `<array of SimpleTypeDescription>`
Type: SimpleTypeDescription object (see 9.2.15)
Requirement: Mandatory

For details see OPC UA specification Annex A: Part 14-A1.5

9.2.5 ConfigurationVersionDataType

The element `configurationVersion` in `ua-data` and `MetaDataVersion` in `ua-metadata` are objects of type `ConfigurationVersionDataType`.

The `ConfigurationVersionDataType` object contains the following elements (see also OPC UA Part 14-6.2.2.1.5-Table 8):

`majorVersion`:

Value range: `<UINT32>`
Type: VersionTime which has TypeOf UInt32
Requirement: Mandatory

Timestamp of metadata definition in seconds since 1st January 2000 (Part 4-7.44).
The *MajorVersion* reflects the time of the last major change of the *DataSet* content.

`minorVersion`:

Value range: `<UINT32>`
Type: VersionTime which has TypeOf UInt32
Requirement: Mandatory

Timestamp of metadata definition in seconds since 1st January 2000 (Part 4-7.44).
The *MinorVersion* reflects the time of the last change.

9.2.6 FieldMetaData

The `FieldMetaData` object contains the following elements (see also OPC UA Part 14-6.2.2.1.3-Table 5):

`name`:

Value range: Unique String representing the name
Type: String
Requirement: Mandatory

Name of the field. The name shall be unique in the `DataSet`.

`description`:

Value range: `<LocalizedText>`
Type: LocalizedText (see 9.2.7)
Requirement: Mandatory

Description of the field.

`fieldFlags`:

Value range: `<DataSetFieldFlags>`
Type: Subtype of UInt16
Requirement: Mandatory

Flags for the field; see definition in OPC UA Part 14-6.2.2.1.4.

`builtInType`:

Value range: `<Byte>`
Type: Byte
Requirement: Mandatory

builtInType values are defined in OPC UA Part 6-5.1.2.

`dataType`:

Value range: `<NodeId>`
Type: NodeId object
Requirement: Mandatory

JSON representation of NodeId is defined in OPC UA Part 6-5.4.2.10-Table 23.

NOTE *First pitfall is to wonder about missing* `Namespace` *in* `dataType` *object. If* `Namespace` *is equal to 0, it is not present in most implementations.*

`valueRank`:

Value range: `<INT32>`
Type: Int32
Requirement: Mandatory

Defines if `dataType` is an array and how many dimensions it has.

`arrayDimensions[]`:

Value range: `<array of UINT32>`
Type: UInt32
Requirement: Mandatory

This field specifies the maximum length of each dimension.

`maxStringLength`:

Value range: `<UINT32>`
Type: UInt32
Requirement: Mandatory

If the `dataType` field is a String or ByteString, this field specifies the maximum length of the String or array.

`dataSetFieldId`:

Value range: `<GUID>`
Type: Guid
Requirement: Mandatory

The unique ID for the field in the dataSet. GUID-Definition: OPC UA Part 6-5.1.3: 16 Byte as JSON string with separator (Part 6-5.4.2.7)

`properties[]`:

Value range: `<array of KeyValuePair>`
Type: KeyValuePair
Requirement: Mandatory

List of Property values providing additional semantics for the field.

9.2.7 LocalizedText

The `LocalizedText` object contains the following elements (see also OPC UA Part 3-8.5-Table 26):

`locale`:

Value range: `<LocaleId>`
Type: String
Requirement: Mandatory

The LocaleId that explicitly identifies the language and the country/region, is represented by two letters in lower case for language and two to three letters in upper case for country, separated by a hyphen (e.g. "en-US" or "en-EN"; see Part 3-8.4).

`text`:

Value range: `<String>`
Type: String
Requirement: Mandatory

The localized text

NOTE *Because of the limitation to one single localized text, we strongly recommend to use English.*

9.2.8 KeyValuePair

The `KeyValuePair` object contains the following elements (see also OPC UA Part 5-12.21-Table 165):

`key`:

Value range: `<QualifiedName>`
Type: QualifiedName
Requirement: Mandatory

This object is defined in Part 3-8.3-Table 24 and contains a namespaceIndex and a name.

`value`:

Valuerange: `<BaseDataType>`
Type: BaseDataType
Requirement: Mandatory

This abstract DataType defines a value that can have any valid DataType (see Part 3-8.7).

9.2.9 StructureDescription

The `StructureDescription` object contains the following elements (see also OPCUA Part 14-A.1.3 Table A.5):

`structureDefinition`:

Value range: `<StructureDefinition>`
Type : StructureDefinition object (see 9.2.10)
Requirement: Mandatory

9.2.10 StructureDefinition

The `StructureDefinition` object contains the following elements (see also OPCUA Part 3-8.49-Table 34):

`defaultEncodingId`:

Value range: `<NodeId>`
Type: NodeId object
Requirement: Mandatory

JSON representation of NodeId is defined in OPC UA Part 6-5.4.2.10-Table 23.

`baseDataType`:

Value range: `<NodeId>`
Type: NodeId object
Requirement: Mandatory

JSON representation of NodeId is defined in OPC UA Part 6-5.4.2.10-Table 23.

`structureType`:

Value range: `<EnumStructureType>`
Type: EnumStructureType
Requirement: Mandatory

`Structure_0`: Structure without optional fields
`StructureWithOptionalFields_1`: Structure with optional fields
`Union_2`: Only one of the fields defined for the data type is encoded into a value

`fields[]`:

Value range: `<array of StructureField>`
Type: StructureField object (see 9.2.11)
Requirement: Mandatory

9.2.11 StructureField

The `StructureField` object contains the following elements (see also OPCUA Part 3-8.51-Table 36):

`name`:

Value range: `<unique name for field in StructureDefinition>`
Type: String
Requirement: Mandatory

`description`:

Value range: `<LocalizedText>`
Type: LocalizedText (see 9.2.7)
Requirement: Mandatory

`datatype`:

Value range: `<NodeId>`
Type: NodeId object
Requirement: Mandatory

JSON representation of NodeId is defined in OPC UA Part 6-5.4.2.10-Table 23.

`valueRank`:

Value range: `<INT32>`
Type: Int32
Requirement: Mandatory

Scalar (-1) or fixed rank Array (>=1)

`arrayDimensions[]`:

Value range: `<array of UINT32>`
Type: UInt32
Requirement: Mandatory

`maxStringLength`:

Value range: `<UINT32>`
Type: UInt32
Requirement: Mandatory

`isOptional`:

Value range: `<BOOLEAN>`
Type: Boolean
Requirement: Mandatory

9.2.12 EnumDescription

The `EnumDescription` object contains the following elements (see also OPCUA Part14-A.1.4-Table A.7):

`enumDefinition`:

Value range: `<EnumDefinition>`
Type: EnumDefinition object (see 9.2.13)
Requirement: Mandatory

`builtInType`:

Value range: `<Byte>`
Type: Byte
Requirement: Mandatory

Indicates whether the DataType is an Enumeration or an OptionSet
6 = Int32 => Enumeration
22 = ExtensionObject => OptionSet
28 = Uinteger => OptionSet

9.2.13 EnumDefinition

The `EnumDefinition` object contains the following elements (see also OPC UA Part 3-8.50-Table 35):

`fields[]`:

Value range: `<array of EnumField>`
Type: EnumField (see 9.2.14)
Requirement: Mandatory

9.2.14 EnumField

The `EnumField` object contains the following elements (see also OPC UA Part 3-8.52-Table 37):

`name`:

Value range: `<Unique name within EnumDefinition>`
Type: String
Requirement: Mandatory

9.2.15 SimpleTypeDescription

The `SimpleTypeDescription` object contains the following elements and can be found in OPC UA Core DataTypes.

`BaseDataType`:

Value range: `<NodeId>`
Type: NodeId
Requirement: Mandatory

`BuiltInType`:

Value range: `<Byte>`
Type: Byte
Requirement: Mandatory

`DataTypeId`:

Value range: `<NodeId>`
Type: NodeId
Requirement:

`Name`:

Value range: `<QualifiedName>`
Type: QualifiedName
Requirement:

9.2.16 ServiceNetworkMessage

The Message Bus payload in call/reply pattern is an object of type ServiceNetworkMessage and is inspired by OPC UA's NetworkMessage, defined in OPC UA Part 14-7.2.3.2 (see 9.2.1 in this document).

The ServiceNetworkMessage object contains the following elements:

`MessageId`:

Value range: `<unixTimestampInMs-PublisherId>`
Type: String
Requirement: Mandatory

Example: `"1567062381000-OTConnector/company.com/type/order/4711"`
Must be unique for any single package of this `PublisherId`.

NOTE *The* `MessageId` *must be unique. The Open Industry 4.0 Alliance defiens the type as a combination of the current timestamp in ms precision with the* `PublisherId`*. In some rare cases, the system timestamp might not be precise enough to avoid sending packages with an unique* `MessageId`*. In these cases, the application must guarantee the uniqueness by providing an additional parameter (example.* `MessageId` = `<unixTimestampInMs><counter>-<PublisherId>`*).*

`MessageType`:

Value range: `"MSG"`
Type: String
Requirement: Mandatory

Only `MSG` is valid here

`PublisherId`:

Value range: `<serviceType>/<appId>`
Type: String consisting of `serviceType` (see 8.1.2) and `oi4Identifier` (see 3.1) - separated by a `/`.

Requirement: Mandatory in Open Industry 4.0 Alliance context, but not in OPC UA context.

`DataSetClassId`:

Value range: `<GUID>` in accordance to OPC UA Part 6-5.1.3: 16 Byte as JSON string with separator (Part 6-5.4.2.7).
Type: String
Requirement: Optional

Example: "`f1875b4a-3209-431b-a38d-2df5758f92c8`"
The `dataSetClassId` allows to refer to the DataSetClass describing the structure of the message. The `dataSetClassId` identifies a well defined `DataSet` specified by the Alliance or some other standards.
For some recurring use cases, such as `newDataSetWriterId`, fixed GUIDs are specified from the Alliance and must be used (A2).

NOTE *The `DataSetClassId` shall be present for all resources defined by the Open Industry 4.0 Alliance, which are having a `DataSetClassId`. This guarantees the availability of a fully schema verifiable messaging.*

`correlationId`:

Value range: `<empty/omitted> or <MessageId>`
Type: String
Requirement: Conditional

Shows the flow between the causal event and its consequences.

NOTE *The `correlationId` is filled in by the first consumer with the `MessageId` of the original message and then passed on from service to service until the message is no longer processed.*

`Message`:

Value range: `<ServiceParametersRequest>` or `<ServiceParametersResponse>`
Type: ServiceParametersRequest object (see 9.2.17) or
ServiceParametersResponse object (see 9.2.19)
Requirement: Mandatory

9.2.17 ServiceParametersRequest

The Open Industry 4.0 Alliance created a ServiceParameterRequest object, which is related to OPC UA its method call service (see OPC UA Part 4-5.11.2.2-Table 65).

The ServiceParametersRequest object contains the following elements:

`methodsToCall[]`:

Value range: `<array of CallMethodRequest>`
Type: Array of CallMethodRequest objects, which describes the messages to call (see 9.2.18)
Requirement: Mandatory

NOTE *The order of execution of methods is up to the receiver.*

9.2.18 CallMethodRequest

The CallMethodRequest object (see OPC UA Part 4-5.11.2.2-Table 65) contains the following elements:

`methodId`:

Value range: `<method name>`
Type: String
Requirement: Mandatory

Name of the method to invoke.

`inputArguments[]`:

Value range: `<array of BaseDataType>`
Type: BaseDataTypes
Requirement: Mandatory

List of input argument values. An empty list indicates that there are no input arguments. The size and order of this list match the size and order of the input arguments defined by the input `inputArguments` property of the method.

9.2.19 ServiceParametersResponse

The Open Industry 4.0 Alliance created a ServiceParameterResponse object, which is related to OPC UA's method call service (see OPC UA Part 4-5.11.2.2-Table 65).

The ServiceParametersResponse object contains the following elements:

`results[]`:

Value range: `<array CallMethodResult>`
Type: CallMethodResult object, which describes the result of the called methods (see 9.2.20)
Requirement: Mandatory

NOTE *The order of method* `results` *must be in the same order as* `methodsToCall` *elements were placed in* `ServiceParameterRequest`*.*

9.2.20 CallMethodResult

The CallMethodResult object (see OPC UA Core-DataTypes) contains the following elements:

`statusCode`:

Value range: `<StatusCode>`
Type: StatusCode, which is a UInt32 and coded as a bit field (see OPC UA Part 4-7.34.1-Table 175)
Requirement: Mandatory

StatusCode of the method executed. This StatusCode is set to the Bad_InvalidArgument if at least one input argument broke a constraint (e.g. wrong data type, value out of range). This StatusCode is set to a bad StatusCode if the method execution failed (e.g. based on an exception).

`inputArgumentResults[]`:

Value range: `<array of StatusCode>`
Type: StatusCode, which is an UInt32 and coded as a bit field (see OPC UA Part 4-7.34.1-Table 175)
Requirement: Mandatory

List of StatusCodes corresponding to the inputArguments. This list is empty unless the operation level result is Bad_InvalidArgument. If this list is populated, it has the same length as the `inputArguments` list.

`outputArguments[]`:

Value range: `<array of BaseDataType>`
Type: BaseDataTypes
Requirement: Mandatory

List of output argument values. An empty list indicates that there are no output arguments. The size and order of this list matches the size and order of the output arguments defined by the `outputArguments` property of the method.

NOTE *The order of* `inputArgumentResults` *must be in the same order as* `inputArguments` *were placed in* `CallMethodRequest`*.*

9.3 OPC UA objects, defined by the Alliance

This chapter explains the most necessary objects and related data types, which the Open Industry 4.0 Alliance are using for pub/sub communication over the Message Bus.

All the objects are JSON encoded and OPC UA pub/sub conform.

9.3.1 mam (Master Asset Model)

The DataSet for Master Asset Model (`mam`) is used by the resource `mam`, which is explained in 8.1.5.1 and used in 10.1.1. The object represents the nameplate of a single asset (device, application, ...). It is derived from IVendorNameplateType, described in OPC UA Part 100 (Part 100-4.5.2).

Because `mam` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a DataSetClassId for this DataSet (A2).

The payload of a DataSetMessage (9.2.3) containing a Master Asset Model is described in detail in section 4.

9.3.2 health

The DataSet `health` is used by the resource `health`, which is explained in 8.1.5.1 and used in 10.1.2. The object represents the actual condition of a single asset (device, application, …). It is derived from IDeviceHealthType, described in OPC UA Part 100 (Part 100-4.5.4), but extended by an additional property called `healthScore`.

Because `health` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a DataSetClassId for this DataSet (A2).

The payload of a DataSetMessage (9.2.3) containing `health` information is structured as followed:

`health`:

Value range: `<DeviceHealthEnumeration>`
Type: DeviceHealthEnumeration has TypeOf enumeration. It is defined for OPC UA-JSON as `<name>_<value>` (OPC UA Part 6-5.4.4).
Access: read only
Requirement: Mandatory

`health` indicates the status as defined by the NAMUR recommendation NE107 and its type is `DeviceHealthEnumeration`(OPC UA Part 100-4.5.4).
The enumeration defines the asset condition, which is described in OPC UA Part 100-4.5.4-Table 22.
The best practice is to combine the five states `NORMAL_0`, `FAILURE_1`, `CHECK_FUNCTION_2`, `OFF_SPEC_3` and `MAINTENANCE_REQUIRED_4` with the symbol/color definitions made by NAMUR NE107 (Table 9).

`healthScore`:

Value range: `<Byte>`
Type: Byte in a range of 0..100 %
Access: read only
Requirement: Optional

`healthScore` reflects a meter to indication a current health level as a result of sub-optimal process and/or environmental conditions in the range from 0 to 100 %.

NOTE *There are no rules, how* `health` *and* `healthScore` *are related to each other. Depending on an asset, its implementation, used technology/protocol/etc. and the use case it is made for, a* `healthScore` *of 30 % can coexist with a* `health` *of NORMAL_0, e.g. when a yearly service has to be done in short term.*

NE107 status	Definition	Color	Symbol
NORMAL_0	Normal operation.	green	
FAILURE_1	Failure (high severity) Signal invalid due to malfunction in the device, sensor or actuator.	red	
CHECK_FUNCTION_2	Function check (low severity) Signal temporarily invalid (e.g. frozen) due to on-going work on the device.	orange	
OFF_SPEC_3	Out of Specification (medium severity) Permissible ambient or process conditions exceeded or the measuring uncertainty of sensors or deviations from the set value in actuators is probably greater than expected.	yellow	
MAINTENANCE_REQUIRED_4	Maintenance required (low severity) Although the signals are valid, the remaining life is nearly exhausted or a function will soon be restricted due to operational conditions.	blue	

Table 13 NAMUR NE107 - Symbol definition

9.3.3 config

The DataSet for `config` is used by the resource `config`, which is explained in 8.1.5.1 and used in 10.1.3. The object represents the current configuration of an asset (device, application, …). Several `config` DataSetMessages may exist for an asset.

Using `config` makes it easy to enable/disable functionalities of an application or configure behaviors such as network settings, scan ranges, etc. over the Message Bus. Through the defined schema of a config DataSet, it is possible to offer generic user interfaces to do this.

Because `config` is defined and referenced by this guideline, the Alliance provides a DataSetClassId for this DataSet (A2).

The complete config object has the same structure for the methods `pub` and `set`, however, they differ in which properties are mandatory. The differences are explained in 9.3.3.1 for `pub` and in 9.3.3.2 for `set`.

9.3.3.1 config (pub)

The payload for the method `pub` of a DataSetMessage (9.2.3) containing `config` information is structured as followed:

`<groupName>`:

Value range: `<group object>`
Type: Object of type `group`
Access: read/write
Requirement: Mandatory

The `group` helps to group configuration elements which should be configured at once, e.g. the group *eth0* might contain configuration objects called *ip_address*, *subnet_mask* and *standard_gateway*.
The group name cannot be changed from outside the service specifying it, but the object group has elements which allow read/write access!

A configuration object can only be added to a `group`. Therefore at least one `group` is mandatory, even when it contains only one single configuration object or several objects, which are not tightly coupled.

Several `group` objects can be added to a single DataSetMessage of type `config`.

It is best practice to name the `group` that collects all common configuration objects *common*. Configuration objects in this group do not necessarily have to be grouped this way, but it is a way to make it clear that these objects belong to the "common settings".

NOTE *To avoid problems on JSON parsing in different languages, only the characters <a..z>, <A..Z>, <0..9>, <-> and <_> are allowed for the* `groupName`*. It is not allowed to start the* `groupName` *with a number (0..9) or with a minus (-).*

`name`:

Value range: `<Localized name>`
Type: LocalizedText
Access: read only
Requirement: Mandatory

A DisplayName for this `group` object shall be given as localized text. The `name` can be used to visualize grouped information such as “ETH0 Settings”, when configuration should be displayed.

In opposite to the object name of the `group`, this `name` is localized. If language support is implemented, the name could be “ETH0 settings” in English or “ETH0 Einstellungen” in German.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`description`:

Value range: `<Localized description>`
Type: LocalizedText
Access: read only
Requirement: Optional

A short `description` of the `group` as localized text can be added here. The `description` can be used for several things, such as displaying it in a generic configuration tool.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`<configName>`:

Value range: `<config object>`
Type: Object of type `config`
Access: read/write
Requirement: Mandatory

The `config` object contains several properties which helps to describe and understand a single configuration item.
The config name cannot be changed from outside the service specifying it, but the object `config` has elements which allow read/write access!

NOTE *To avoid problems on JSON parsing in different languages, only the characters <a..z>, <A..Z>, <0..9>, <-> and <_> are allowed for the* `configName`*. It is not allowed to start the* `configName` *with a number (0..9) or with a minus (-).*

`type`:

Value range: `<Subset of OPC UA Base Types>`
Type: String
Access: read only
Requirement: Mandatory

A subset of OPC UA base types, defined in OPC UA Part 5-12.2-Table 116 and more detailed in Part 3-8 shall be used as `type`.
The allowed subset is Boolean, ByteString, DateTime, Number and String.

Each type will be presented as string in value key (e.g. a DateTime will be a stringified object).

If a subset of a data type (e.g. the value range of an UINT16 is -100…+1200) is used, it can be specified with help of `validation` object.

NOTE *Via use of the* `validation` *key* `pattern`*, it is possible to define arbitrary data types if needed.*

`value`:

Value range: `<value>`
Type: String
Access: read/write
Requirement: Mandatory

The `value` of a configuration item is always a string, which needs to be interpreted with help of the `type` information.

NOTE *To delete, or better unset, a configuration, the related value must be overwritten with an empty string.*

`unit`:

Value range: `<unit>`
Type: String
Access: read only
Requirement: Optional

The `unit` should contain the “DisplayName” of one of the EngineeringUnits defined in OPC UA Part 8-5.6.3 (see UNECE Recommendation N° 20 for a good overview).

However, when a very rare unit does not exists in the list of EngineeringUnits, it is allowed to use a self defined (or better industry specific) unit here.

`defaultValue`:

Value range: `<value>`
Type: String
Access: read only
Requirement: Optional

The `defaultValue` of a configuration item is always a string, which needs to be interpreted with help of the `type` information.

`mandatory`:

Value range: `<true/false>`
Type: Boolean
Access: read only
Requirement: Optional

If `mandatory` is set to true, this `config` object must be part of the `group` and `value` must be set during write operation.

This property is optional. If it does not exists, the default behavior is the same as `mandatory` = `false`.

`sensitive`:

Value range: `<true/false>`
Type: Boolean
Access: read only
Requirement: Optional

In context with the available OPC UA base types the Alliance lacks a type "password" which can be used to hide the value of an object (e.g. in user interfaces, such as web front ends) if necessary. The property `sensitive` is used to fulfill these needs.

This property is optional. If it does not exists, the default behavior is the same as `sensitive` = `false`.

NOTE *The property* `sensitive` *is an indicator for front end applications to handle this value with a special manor. The* `value` *itself contains a standard string.*

`name`:

Value range: `<LocalizedText>`
Type: LocalizedText
Access: read only
Requirement: Mandatory

A DisplayName for this `config` object shall be given as localized text. It can be used to visualize the `name` such as “IP Address”, in a specific language.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`description`:

Value range: `<LocalizedText>`
Type: LocalizedText
Access: read only
Requirement: Optional

A short `description` of this `config` object as localized text can be added here. The `description` can be used for several things, such as displaying it in a generic configuration tool.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`validation`:

Value range: `<validation object>`
Type: Object of type validation
Access: read only
Requirement: Optional

The `validation` object helps to verify the value of the `config` object.

`length`:

Value range: `<UINT32>`
Type: UInt32
Access: read only
Requirement: Optional

The `length` contains the max length of a `value`.

`min`:

Value range: `<REAL>`
Type: Real
Access: read only
Requirement: Optional

The `min` contains the minimal value, a `value` can get.

`max`:

Value range: `<REAL>`
Type: Real
Access: read only
Requirement: Optional

The `max` contains the maximum value, a `value` can get.

`pattern`:

Value range: `<string>`
Type: String
Access: read only
Requirement: Optional

The `pattern` might contain a regular expression to check the content of a `value`.

NOTE *Be aware, that programming language specific flavors for regular expressions exist. Therefore, it is encouraged to specify interoperable patterns only.*

`values[]`:

Value range: `<array of strings>`
Type: String
Access: read only
Requirement: Optional

A type of enumeration to show, which values are allowed.

`context`:

Value range: `<Context object>`
Type: Object of type `context`
Access: read/write
Requirement: Optional

The `context` helps to define an overall name for the included `group` objects. E.g. the `group` objects with the name “eth0”, “eth1” and “eth2” are all related to “Network settings”, which is the `context`. The `context` (e.g. “Network settings”) is related to the Filter of the DataSetMessage, which might be “Network%20settings”.

`name`:

Value range: `<Localized name>`
Type: LocalizedText
Access: read only
Requirement: Mandatory

A DisplayName for the `context` of the involved `group` objects as localized text. The `name` can be used to visualize `context` information such as “Network Settings”, when configuration should be displayed.

In opposite to the object name of the group, this `name` is localized. If language support is implemented, the name could be “Network settings” in English or “Netzwerk Einstellungen” in German.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`description`:

Value range: `<Localized description>`
Type: LocalizedText
Access: read only
Requirement: Optional

A short `description` of the `context` as localized text can be added here. The `description` can be used for several things, such as displaying it in a generic configuration tool.

NOTE *Details on how to get localized information are described in section 9.3.14.*

9.3.3.2 config (set)

The payload for the method `set` of a DataSetMessage (see 9.2.3) containing `config` information is structured as followed:

`<groupName>`:

Value range: `<group object>`
Type: Object of type `group`
Access: read/write
Requirement: Mandatory

The `group` helps to group configuration elements which should be configured at once, e.g. the group *eth0* might contain configuration objects called *ip_address*, *subnet_mask* and *standard_gateway*.
The group name cannot be changed from outside the service specifying it, but the object group has elements which allow read/write access!

A configuration object can only be added to a `group`. Therefore at least one `group` is mandatory, even when it contains only one single configuration object or several objects, which are not tightly coupled.

Several `group` objects can be added to a single DataSetMessage of type `config`.

It is best practice to name the `group` that collects all common configuration objects *common*. Configuration objects in this group do not necessarily have to be grouped this way, but it is a way to make it clear that these objects belong to the "common settings".

NOTE *To avoid problems on JSON parsing in different languages, only the characters*

<a..z>, <A..Z>, <0..9>, <-> and <_> are allowed for the `groupName`*. It is not allowed to start the* `groupName` *with a number (0..9) or with a minus (-).*

`name`:

Value range: `<Localized name>`
Type: LocalizedText
Access: read only
Requirement: Optional

A DisplayName for this `group` object shall be given as localized text. The `name` can be used to visualize grouped information such as “ETH0 Settings”, when configuration should be displayed.

In opposite to the object name of the `group`, this `name` is localized. If language support is implemented, the name could be “ETH0 settings” in English or “ETH0 Einstellungen” in German.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`description`:

Value range: `<Localized description>`
Type: LocalizedText
Access: read only
Requirement: Optional

A short `description` of the `group` as localized text can be added here. The `description` can be used for several things, such as displaying it in a generic configuration tool.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`<configName>`:

Value range: `<config object>`
Type: Object of type `config`
Access: read/write
Requirement: Mandatory

The `config` object contains several properties which helps to describe and understand a single configuration item.
The config name cannot be changed from outside the service specifying it, but the object `config` has elements which allow read/write access!

NOTE *To avoid problems on JSON parsing in different languages, only the characters <a..z>, <A..Z>, <0..9>, <-> and <_> are allowed for the* `configName`*. It is not allowed to start the* `configName` *with a number (0..9) or with a minus (-).*

`type`:

Value range: `<Subset of OPC UA Base Types>`
Type: String
Access: read only
Requirement: Optional

A subset of OPC UA base types, defined in OPC UA Part 5-12.2-Table 116 and more detailed in Part 3-8 shall be used as `type`.
The allowed subset is Boolean, ByteString, DateTime, Number and String.

Each type will be presented as string in value key (e.g. a DateTime will be a stringified object).

If a subset of a data type (e.g. the value range of an UINT16 is -100…+1200) is used, it can be specified with help of `validation` object.

NOTE *Via use of the* `validation` *key* `pattern`*, it is possible to define arbitrary data types if needed.*

`value`:

Value range: `<value>`
Type: String
Access: read/write
Requirement: Mandatory

The `value` of a configuration item is always a string, which needs to be interpreted with help of the `type` information.

NOTE *To delete, or better unset, a configuration, the related value must be overwritten with an empty string.*

`unit`:

Value range: `<unit>`
Type: String
Access: read only
Requirement: Optional

The `unit` should contain the “DisplayName” of one of the EngineeringUnits defined in OPC UA Part 8-5.6.3 (see UNECE Recommendation N° 20 for a good overview).

However, when a very rare unit does not exists in the list of EngineeringUnits, it is allowed to use a self defined (or better industry specific) unit here.

`defaultValue`:

Value range: `<value>`
Type: String
Access: read only
Requirement: Optional

The `defaultValue` of a configuration item is always a string, which needs to be interpreted with help of the `type` information.

`mandatory`:

Value range: `<mandatory>`
Type: Boolean
Access: read only
Requirement: Optional

If `mandatory` is set to true, this `config` object must be part of the `group` and `value` must be set during write operation.

This property is optional. If it does not exists, the default behavior is the same as `mandatory` = `false`.

`sensitive`:

Value range: `<sensitive>`
Type: Boolean
Access: read only
Requirement: Optional

In context with the available OPC UA base types the Alliance lacks a type "password" which can be used to hide the value of an object (e.g. in user interfaces, such as web front ends) if necessary. The propery `sensitive` is used to fulfill these needs.

This property is optional. If it does not exists, the default behavior is the same as `sensitive` = `false`.

NOTE *The property* `sensitive` *is an indicator for front end applications to handle this value with a special manor. The* `value` *itself contains a standard string.*

`name`:

Value range: `<LocalizedText>`
Type: LocalizedText
Access: read only
Requirement: Optional

A DisplayName for this `config` object shall be given as localized text. It can be used to visualize the `name` such as “IP Address”, in a specific language.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`description`:

Value range: `<LocalizedText>`
Type: LocalizedText
Access: read only
Requirement: Optional

A short `description` of this `config` object as localized text can be added here. The `description` can be used for several things, such as displaying it in a generic configuration tool.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`validation`:

Value range: `<validation object>`
Type: Object of type `validation`
Access: read only
Requirement: Optional

The `validation` object helps to verify the value of the `config` object.

`length`:

Value range: `<UINT32>`
Type: UInt32
Access: read only
Requirement: Optional

The `length` contains the max length of a `value`.

`min`:

Value range: `<REAL>`
Type: Real
Access: read only
Requirement: Optional

The `min` contains the minimal value, a `value` can get.

`max`:

Value range: `<REAL>`
Type: Real
Access: read only
Requirement: Optional

The `max` contains the maximum value, a `value` can get.

`pattern`:

Value range: `<string>`
Type: String
Access: read only
Requirement: Optional

The `pattern` might contain a regular expression to check the content of a `value`.

NOTE *Be aware, that programming language specific flavors for regular expressions exist. Therefore, it is encouraged to specify interoperable patterns only.*

`values[]`:

Value range: `<array of strings>`
Type: String
Access: read only
Requirement: Optional

A type of enumeration to show, which values are allowed.

`context`:

Value range: `<Context object>`
Type: Object of type `context`
Access: read/write
Requirement: Optional

The `context` helps to define an overall name for the included `group` objects. E.g. the `group` objects with the name “eth0”, “eth1” and “eth2” are all related to “Network settings”, which is the `context`. The `context` (e.g. “Network settings”) is related to the Filter of the DataSetMessage, which might be “Network%20settings”.

`name`:

Value range: `<Localized name>`
Type: LocalizedText
Access: read only
Requirement: Mandatory

A DisplayName for the `context` of the involved `group` objects as localized text. The `name` can be used to visualize `context` information such as “Network Settings”, when configuration should be displayed.

In opposite to the object name of the group, this `name` is localized. If language support is implemented, the name could be “Network settings” in English or “Netzwerk Einstellungen” in German.

NOTE *Details on how to get localized information are described in section 9.3.14.*

`description`:

Value range: `<Localized description>`
Type: LocalizedText
Access: read only
Requirement: Optional

A short `description` of the `context` as localized text can be added here. The `description` can be used for several things, such as displaying it in a generic configuration tool.

NOTE *Details on how to get localized information are described in section 9.3.14.*

9.3.4 license

The DataSet `license` is used by the resource `license`, which is explained in 8.1.5.1 and used in 10.1.4. The object represents the actual license information for a single application.

NOTE *To be compliant with the license, many licenses require to state out the terms of the license agreement. The resource* `license` *lists all relevant licenses.*

Because `license` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a `DataSetClassId` for this `DataSet` (A2).

The payload of a DataSetMessage (9.2.3) containing `license` information is structured as followed:

`components[]`:

Value range: `<array of componentsObject>`
Type: `componentsObject`
Access: read only
Requirement: Mandatory

The `components` list contains all software components, which are licensed under the same license. Both, the DataSetWriterId and the Filter from the DataSetMessage header are related to that license.

The `componentsObject` is structured as following:

`component`:

Value range: `<component name>`
Type: String
Access: read only
Requirement: Mandatory

This is the name of the `component`, which uses the license names in the father object.

`licAuthors[]`:

Value range: `<array of name of author>`
Type: String
Access: read only
Requirement: Optional (not present, if no author information are available)

This is a list of authors, which are providing this `component`.

`licAddText`:

Value range: `<additional license text>`
Type: String
Access: read only
Requirement: Optional

Some components may have extended license information in addition to the license agreements defined for it. This additional information can be placed in `licAddText` as plain text.

9.3.5 licenseText

The DataSet for `licenseText` is used by the resource `licenseText`, which is explained in 8.1.5.1 and used in 10.1.5. The object represents the actual license text for a specific `licenseId` for a single application.

NOTE *To be compliant with the license, many licenses require to state out the terms of the license agreement. The resource* `licenseText` *lists the relevant information.*

Because `licenseText` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a `DataSetClassId` for this `DataSet` (A2).

The payload of a DataSetMessage (9.2.3) containing `licenseText` information is structured as followed:

`licenseText`:

Value range: `<license text>`
Type: String
Access: read only
Requirement: Mandatory

The defined license text for the given `licenseId` will be placed in `licenseText` as plain text.

9.3.6 rtLicense

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The DataSet for `rtLicense` is used by the resource `rtLicense`, which is explained in 8.1.5.1 and used in 10.1.6. The object represents the actual runtime license(s) information for a single application.

Because `rtLicense` is defined and referenced by this guideline, the Alliance provides a DataSetClassId for this DataSet (A2).

The payload of a DataSetMessage (9.2.3) containing `rtLicense` information is structured as followed:

<not yet defined>

9.3.7 data

The DataSet for `data` is used by the resource `data`, which is explained in 8.1.5.1 and used in 10.1.7. Applications can offer DataSets "on their own", and pack them into DataSetMessages, which are then available via resource `data`.

The payload of a DataSetMessage (9.2.3) containing `data` is not defined by the Alliance. Each data object might look different, but it has to follow the rules of OPC Foundation's specification for JSON coded pub/sub DataSets (OPC UA Part 14-7.2.3.3-Table 92). These DataSetMessages are likely to be generated based on an existing device description or domain specific knowledge.

The Alliance defines an optional DataSet to access the available process values in a standard way. This DataSetMessage is called oi4_pv and its key-value-pairs following a fixed key structure, i.e. its key elements are defined as below.

NOTE *It is published via .../pub/data/<oi4Identifier>/oi4_pv.*

NOTE *The oi4_pv DataSet follows the idea of process values in the process industry, containing "Primary Value" and “Secondary Values”.*

The payload of a DataSetMessage (9.2.3) containing `data` in oi4_pv format is structured as followed:

`pv`:

Value range: `<any>`
Type: any
Access: read/write (application defined)
Requirement: Mandatory

`pv` indicates the primary value of an asset. E.g., it could be a temperature value for a temperature sensor.
Unit, range and all other metadata are accessible via the resource `metadata`.
Depending on the type of `pv`, the value can be a scalar, an array of values or a complex object.

NOTE *The reflected measuring point, most likely defined by a device description of the manufacturer, should be placed into the "description" field of the related metadata.*

`sv_<n>`:

Value range: `<any>`
Type: any
Access: read/write (application defined)
Requirement: Mandatory

`sv_<n>` indicates a secondary value of an asset. E.g., it could be a humidity value for a temperature sensor.
Unit, range and all other metadata are accessible via the resource `metadata`.

Depending on the type of `pv`, the value can be a scalar, an array of values or a complex object.

NOTE *The range of secondary values are 1 to 255.*

NOTE *The reflected measuring point, most likely defined by a device description of the manufacturer, should be placed into the "description" field of the related metadata.*

9.3.8 metadata

The DataSetMetaData for `metadata` is used by the resource `metadata`, which is explained in 8.1.5.1 and used in 10.1.8. For a better clarification, the `metadata` is of type DataSetMetaData (9.2.2) and contains the metadata to a defined DataSet. The DataSet, which is embedded in a DataSetMessage (9.2.3) is conform to OPC UA Part 14 and related to Alliance' resource `data`, which are described in 9.3.7.

9.3.9 event

The DataSet for `event` is used by the resource `event`, which is explained in 8.1.5.1 and used in 10.1.9. The object represents notifications such as wire-break, out-of-specification warnings (e.g. NE107) or errors, but also information about informing events like a sensor exchange or a user login. The `event` object represents information from both physical devices and applications.

An `event` might be used in other contexts too – e.g. application-specific. For this, the payload object is freely configurable.

The `event` object provides category-specific interpretations, which are explained in the following subsections. This makes it possible to select the most suitable assignment for the specific use case. In addition, category-specific filtering on relevant events is possible and will be explained in 10.1.9.

Because `event` is defined and referenced by this guideline, the Alliance provides a DataSetClassId for this DataSet (A2).

The payload of a DataSetMessage (9.2.3) containing `event` information is structured as followed:

`origin`:

Value range: `<oi4Identifier>`
Type: String
Access: read only
Requirement: Mandatory

The `oi4Identifier` (3) identifies the asset (might be a device or an application), where the event comes from.

`number`:

Value range: `<UINT32>`
Type: UInt32
Access: read only
Requirement: Mandatory

An number to hint to the reason of the event.

`description`:

Value range: `<description>`
Type: String
Access: read only
Requirement: Optional

Human readable `description` for the event `number`.

`category`:

Value range: `<<categoryName>_<categoryValue>>`
Type: enumeration
Access: read only
Requirement: Mandatory

The used category to interpret the following payload.
CAT_SYSLOG_0: Interpret number, description and payload as syslog entry.
CAT_STATUS_1: Interpret number, description and payload as OI4 status
CAT_NE107_2: Interpret number, description and payload as Namur NE107
CAT_GENERIC_99: For everything, which has no standardized schema.

`details`:

Value range: `<category defined object>`
Type: Object
Access: read only
Requirement: Optional

This object depends on the category and its defined schema. In case of usage of CAT_GENERIC_99, no common schema validation is possible.

9.3.9.1 status

1tatus events are according to the following definition.

The element `category` for status events is `CAT_STATUS_1`.

The Open Industry 4.0 Alliance defines to use OPC Status Code information to publish the status of modification requests over the Message Bus as following:

`origin`:

Value range: `<oi4Identifier>`
Type: String
Access: read only
Requirement: Mandatory

The `oi4Identifier` (3) identifies the asset (might be a device or an application), where the event comes from.

`number`:

Value range: `<UINT32>`
Type: UInt32
Access: read only
Requirement: Mandatory

An number which contains the OPC Status Code to hint to the reason and the status of the event (e.g. 0x80060000 for “BadEncodingError“).

`description`:

Value range: `<description>`
Type: String
Access: read only
Requirement: Optional

Human readable `description` for the OPC UA Status Code used in `number` (e.g. “Encoding halted because of invalid data in the objects being serialized.“ for 0x80060000).

`category`:

Value range: `CAT_STATUS_1`
Type: enumeration
Access: read only
Requirement: Mandatory

The used category is always `CAT_STATUS_1`.

`details`:

Value range: `<object>`
Type: Object

Access: read only
Requirement: Optional

`symbolicId`:

Value range: `<Symbolic Id>`
Type: String
Access: read only
Requirement: Optional

A symbolic id, defined by the OPC Foundation, as short form of the description (e.g. “BadEncodingError” for 0x80060000).

NOTE *The event category* `status` *must be used to respond to modification requests, done over the Message Bus.*

NOTE *The event category* `status` *can be used to publish application specific information such as internal state.*

9.3.9.2 syslog

A syslog event provides information as described in RFC3164.

The element `category` for syslog events is `CAT_SYSLOG_0`.

The Alliance defines the syslog event mapping as following:

`origin`:

Value range: `<oi4Identifier>`
Type: String
Access: read only
Requirement: Mandatory

The `oi4Identifier` (3) identifies the asset (might be a device or an application), where the event comes from.

`number`:

Value range: `<PRI>`
Type: UInt32
Access: read only
Requirement: Mandatory

A number which represents the PRI of a syslog message.

The PRI is a 8 bit integer. It consists of Severity and Facility. Three bits are used for the coding of Severity, 5 bits are used for coding Facility.

`category`:

Value range: `CAT_SYSLOG_0`
Type: enumeration
Access: read only
Requirement: Mandatory

The used category is always `CAT_SYSLOG_0`.

`details`:

Value range: `<object>`
Type: Object
Access: read only
Requirement: Optional

`MSG`:

Value range: `<MSG>`
Type: String
Access: read only
Requirement: optional

The `MSG` contains the MSG part of a syslog message.

`HEADER`:

Value range: `<HEADER>`
Type: String
Access: read only
Requirement: optional

The `HEADER` contains the HEADER part of a syslog message, which consists of the name or ip address of the sender and timestamp.

The optional key `description` is not used in context of this `category`.

The `eventLevel` depends on severity level of `PRI`.

9.3.9.3 NAMUR NE107

NAMUR NE107 events are according to the following definition.

The element `category` for NAMUR NE107 events is `CAT_NE107_2`.

The NAMUR NE107 standard describes the status signals 1-4. The Alliance, such as the OPC Foundation in Part 100-4.5.4-Table 22, added the signal 0 that represents a normal operation.

NE107 status	OPC UA Part 100 status	Status signal	Definition	Color	Symbol
0	NORMAL_0	Normal Operation	Normal operation.	green	
1	FAILURE_1	Failure	Failure (high severity) Signal invalid due to malfunction in the device, sensor or actuator.	red	
2	CHECK_FUNCTION_2	Check Function	Function check (low severity) Signal temporarily invalid (e.g. frozen) due to on-going work on the device.	orange	
3	OFF_SPEC_3	Out of Specification	Out of specification (medium severity) Permissible ambient or process conditions exceeded or the measuring uncertainty of sensors or deviations from the set value in actuators is probably greater than expected.	yellow	
4	MAINTENANCE_REQUIRED_4	Maintenance Required	Maintenance required (low severity) Although the signals are valid, the remaining life is nearly exhausted or a function will soon be restricted due to operational conditions.	blue	

Table 14 Alliance NAMRU NE107 status signals

The payload of the event can contain additional information about the event, e.g. to provide failure codes for causes and remedies.

The payload of a DataSetMessage (9.2.3) containing `event` information is structured as followed:

`origin`:

Value range: `<oi4Identifier>`
Type: String
Access: read only
Requirement: Mandatory

The `oi4Identifier` (3) identifies the asset (might be a device or an application), where the event comes from.

`number`:

Value range: `<0 to 4>`
Type: UInt32
Access: read only
Requirement: Mandatory

A number between 0-4 depending on the NE107 status from table 4.

`description`:

Value range: `<description>`
Type: String
Access: read only
Requirement: Optional

Human readable `description` for the NE107 status code used in `number`.

`category`:

Value range: `CAT_NE107_2`
Type: enumeration
Access: read only
Requirement: Mandatory

The used category is always `CAT_NE107_2`.

`details`:

Value range: `<object>`
Type: Object
Access: read only
Requirement: Optional

`diagnosticCode`:

Value range: `<event condition>`
Type: String
Access: read only
Requirement: Optional

Manufacturer specific detail information about the the event. E.g. F-238.

`location`:

Value range: `<location within the asset/service>`
Type: String
Access: read only
Requirement: Optional

Actual location of the raised event. E.g. if an asset has multiple sensor units build like temperature.

9.3.9.4 generic

A generic provides information which are undefined out of the scope of the Open Industry 4.0 Alliance.

The element `category` for generic events is `CAT_GENERIC_99`.

The Open Industry 4.0 Alliance defines the generic event mapping as follows:

`origin`:

Value range: `<oi4Identifier>`
Type: String
Access: read only
Requirement: Mandatory

The `oi4Identifier` (3) identifies the asset (might be a device or an application), where the event comes from.

`number`:

Value range: `<UINT32>`
Type: UInt32
Access: read only
Requirement: Mandatory

A number which represents something, the application should have defined and documented.

`description`:

Value range: `<description>`
Type: String
Access: read only
Requirement: Optional

Human readable `description` for the event `number`.

`category`:

Value range: `CAT_GENERIC_99`
Type: enumeration
Access: read only
Requirement: Mandatory

The used category is always `CAT_GENERIC_99`.

`details`:

Value range: `<object>`
Type: Object
Access: read only
Requirement: Optional

An application defined object which represents something in scope of the application. It should be well documented on application side.

9.3.10 profile

The DataSet for `profile` is used by the resource `profile`, which is explained in 8.1.5.1 and used in 10.1.10. The object represents the actual available resources in a single application or device. It contains an array of all mandatory and optional resources (8.1.5) which are supported.

Because `profile` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a `DataSetClassId` for this `DataSet` (A2).

The payload of a DataSetMessage (9.2.3) containing `profile` information is structured as followed:

`resource[]:`

Value range: `<array of resources>`
Type: String
Access: read only
Requirement: Mandatory

List of all resources an asset can serve.

9.3.11 publicationList

The DataSet for `publicationList` is used by the resource `publicationList`, which is explained in 8.1.5.1 and used in 10.1.11. The object represents the actual publications available in a single application and their rudimentary settings. This includes all publications of the application itself as well as all DataSetMessages provided by underlying devices.

The `publicationList` contains an array of DataSetMessage objects, each representing an available publication with its resource, unique DataSetWriterId, Filter and several configuration settings.

Because `publicationList` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a DataSetClassId for this DataSet (A2).

The payload of a DataSetMessage (9.2.3) containing `publicationList` object is structured as followed:

`resource`:

Value range: `<resource>`
Type: String
Access: read only
Requirement: Mandatory

Name of the `resource` (8.1.5), which is used for the related DataSetMessage (e.g. data, mam, health, ...).

`subResource`:

Value range: `<sub resource>`
Type: String
Access: read only
Requirement: Optional

Name of the `subResource` (8.1.6), which is used for the related DataSetMessage. The `subResource` contains useful information for effective filtering of the available publications. For the `resource` event, for example, the `subResource` contains the

associated event category (syslog, opcSC, ne107, status or generic).

NOTE For most resources, the `subResource` contains the oi4Identifier to identify the associated asset. In these cases, the `subResource` would contain redundant information and is therefore unnecessary.

`filter`:

Value range: `<filter>`
Type: String
Access: read only
Requirement: Optional

Filter name of a dataSet from the application or the underlying device. The `filter` (8.1.7) should have a "speaking" name and is defined by the publishing application, a device description file or from other sources. The `filter`, in combination with the `resource` and `subResource` must be unique within the scope of a publisher. Several `resources` such as man or health does not make usage of `filter`.

NOTE *Since the `filter` may be referenced within a topic, it must not contain special characters. If special characters can occur within the application, it would be possible to avoid problems by URL encoding.*

`DataSetWriterId`:

Value range: `<UINT16>`
Type: UInt16
Access: read only
Requirement: Mandatory

An identifier for DataSetWriter which published the DataSetMessage and DataSetMetaData. The `DataSetWriterId` (9.2.3) is unique within the scope of a publisher, therefore the publisher should assign it.

`oi4Identifier`:

Value range: `<oi4Identifier>`
Type: String
Access: read only
Requirement: Mandatory

The `oi4Identifier` (3) identifies the asset (might be a device or an application), where the dataSet comes from.

`mode`:

Value range: `<<enumName>_<enumValue>>`
Type: enumeration. It is defined for OPC UA-JSON as `<name>_<value>` (OPC UA Part 6-5.4.4).
Access: read/write
Requirement: Optional (`mode` = `ON_REQUEST_1` if not present)

A DataSetMessage may be available in an application/device, but the way it should be published may differ depending on the use case. Different types of publishing are possible within the Alliance.

`OFF_0`:

With `OFF_0` the DataSetMessage is not accessible via the Message Bus and is not published at all. This mode can be used when a DataSetMessage is not relevant for the driven use cases.

`ON_REQUEST_1`:

With `ON_REQUEST_1` the DataSetMessage can be requested via the Message Bus with `oi4/<serviceType>/<appId>/get/<resource>/<subResource>/<filter>`.
The DataSetMessage is not published in any way other than by a request.
This mode may be used when a DataSetMessage is not relevant to the driven use case, but may be of interest to acyclic requests.

`APPLICATION_2`:

With `APPLICATION_2` the DataSetMessage will be published with other DataSetMessages of the same resource (8.1.5) and same application (8.1.3).
For example, all DataSetMessages for the resource mam known by an application are published together in a single NetworkMessage via `oi4/<serviceType>/<appId>/pub/mam`.
Additionally the single DataSetMessage can be requested via the Message Bus.

`SUBRESOURCE_3`:

With `SUBRESOURCE_3` the DataSetMessage will be published with other DataSetMessages of the same resource (8.1.5), same subResource (8.1.6) and same application (8.1.3).
For example, all DataSetMessages for the resource data known by a particular asset are published together in a single NetworkMessage via `oi4/<ServiceType>/<AppId>/pub/data/<oi4Identifier>`.
Additionally the single DataSetMessage can be requested via the Message Bus.

`FILTER_4`:

With `FILTER_4` the DataSetMessage will be published separately in an own NetworkMessage.
For example, a DataSetMessages named "temperature" for the resource data belonging to a particular asset is published to a NetworkMessage via `oi4/<ServiceType>/<AppId>/pub/data/<oi4Identifier>/temperature`.
Additionally the single DataSetMessage can be requested via the Message Bus.

`APPLICATION_SUBRESOURCE_5`:

Enables publishing via the modes `APPLICATION_2` and `SUBRESOURCE_3`.

`APPLICATION_FILTER_6`:

Enables publishing via the modes `APPLICATION_2` and `Filter_4`.

`SUBRESOURCE_FILTER_7`:

Enables publishing via the modes `SUBRESOURCE_3` and `FILTER_4`.

`APPLICATION_SUBRESOURCE_FILTER_8`:

Enables publishing via the modes `APPLICATION_2`, `SUBRESOURCE_3` and `Filter_4`.

NOTE *Be aware, metadata is not a DataSetMessage, but a DataSetMetaDataType. Therefore only FILTER_4 messaging is supported, because a summary mechanism does not exist for metadata.*

`interval`:

Value range: `0…<n>`
Type: UInt32
Access: read/write
Requirement: Optional (`interval`= 0 if not present)

The publishing interval might be set between 0 (immediately on change) and <n> ms (if supported).
By default, any dataSet gets published on change => interval = 0.
If set to > 0 ms it gets published after the interval has expired, regardless of whether a change in value has occurred in the meantime.

NOTE *In combination with* `precisions` *is unequal to 0 and* `interval` *> 0 ms, the value gets published when* `interval` *has expired - regardless if minimum deviation has reached. Therefore* `precisions` *gets ignored, when* `interval` *> 0 ms.*

`precisions`:

Value range: `<precisionObject>`
Type: object
Access: read/write
Requirement: Optional (all `precisions` = 0 (publish on change) if not present)

Each entry of the precisionObject defines the minimum deviation (+/-), a specific value of a given dataSet should have, before it gets published again.
This is useful for floating analog values, e.g. when only value changes >= 0.2 digits should be published.
0 means publish every value change - `precisions` is switched off.
> 0.0 means publish again only when the set delta occurs.
The `precisionObject`, which contains the precision settings for a set of members of a dataSet is defined as:

`<name of dataSet member>`:

Value range: `0.0 ... 3.4·1038`
Type: Real
Access: read/write
Requirement: Mandatory

NOTE `precisions` *is used to reduce unnecessary traffic on the Message Bus. This function can be used if publishing with fixed intervals does not seem to be the right solution.*

`config`:

Value range: `<<enumName>_<enumValue>>`
Type: enumeration. It is defined for OPC UA-JSON as `<name>_<value>` (OPC UA Part 6-5.4.4).
Access: read only
Requirement: Optional (`config`= `NONE_0` if not present)

The configurability of several DataSetMessages is likely different – depending on implementation and/or technical needs:

`NONE_0`:

No configuration possible.

`MODE_1`:

The publishing behavior can be configured between different modes.

`INTERVAL_2`:

Publishing `interval` can be set between 0..<n> ms via interval or `precisions` can be used for numerical data, when republishing should occur only after a minimum defined change.

`MODE_AND_INTERVAL_3`:

Publishing behavior can be set by manipulating `mode`, `interval` and `precisions`.

NOTE *The publicationList is the only place, where all this cross referencing information are available at once.*

9.3.12 subscriptionList

The DataSet for `subscriptionList` is used by the resource `subscriptionList`, which is explained in 8.1.5.1 and used in 10.1.12. The object represents the actual available/configured subscriptions in a single application and its rudimentary settings.

The `subscriptionList` contains an array of DataSetMessage objects, each representing a subscription and contains a topic path an interval for application internal usage and its configurability.

Because `subscriptionList` is defined and referenced by this guideline, the Alliance provides a DataSetClassId for this DataSet (A2).

In combination with a `resourceType` (8.1.7) it is possible to get a `subscriptionList` filtered for a defined resource.

The payload of a DataSetMessage (9.2.3) containing `subscriptionList` object is structured as followed:

`topicPath`:

Value range: `<topic>`
Type: String
Access: read/write
Requirement: Mandatory

`interval`:

Value range: `0…<n>`
Type: UINT32
Access: read/write
Requirement: Optional (`interval`= 0 if not present)

The publishing interval might be set between 0 (immediately on change) and <n> ms (if supported).
By default, any DataSetMessage gets published on change => interval = 0.

NOTE *The subscribing client uses the key* `interval` *to decide, if every publication needs to be computed or not.*
E.g. a value gets published every 20 ms, but in the SubscriptionList of a Cloud connector `interval` *is set to 1000 ms - this means, the cloud connector computes the published data only every 1000 ms, instead of every 20 ms to save bandwidth and costs.*

NOTE *The maximum value of 0xFFFFFFFF represents a recomputing interval from around 50 days.*

`config`:

Value range: `<<enumName>_<enumValue>>`
Type: `enumeration`. It is defined for OPC UA-JSON as `<name>_<value>` (OPC UA Part 6-5.4.4).
Access: read/write
Requirement: Optional (`config` = NONE_0 if not present)

The configurability of subscriptions is limited to create/delete/not manipulable – depending on implementation and/or technical needs:
NONE_0: No configuration possible (delete is not possible).
CONF_1: Free configurable (delete is possible)

9.3.13 interfaces

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The DataSet for `interfaces` is used by the resource `interfaces`, which is explained in 8.1.5.1 and used in 10.1.13. The object represents the physically available interfaces such as connectors, switches, leds for a single device.

Because `interfaces` is defined and referenced by this guideline, the Alliance provides a DataSetClassId for this DataSet (A2).

The payload of a DataSetMessage (9.2.3) containing `interfaces` information is structured as followed:

<not yet defined>

9.3.14 referenceDesignation

The DataSet for `referenceDesignation` is used by the resource `referenceDesignation`, which is explained in 8.1.5.1 and used in 10.1.14. The object represents the actual reference designation of a single asset according to IEC 81346-1:2009-07.

Because `referenceDesignation` is defined and referenced by this guideline, the Open Industry 4.0 Alliance provides a `DataSetClassId` for this `DataSet` (A2).

The payload of a DataSetMessage (9.2.3) containing `referenceDesignation` information is structured as followed:

`function`:

Value range: `<functionObject>`
Type: Object
Access: read/write
Requirement: Optional

The `function` object contains the function oriented reference. A function describes the task or operation that is performed.

The `functionObject` is structured as following:

`value`:

Value range: `<designation value>`
Type: String
Access: read/write
Requirement: Mandatory

This is the function oriented reference designation value.

`local`:

Value range: `<local designation value>`
Type: String
Access: read/write
Requirement: Optional

This is the local part of the the function oriented reference.

`parent`:

Value range: `<functionParent>`
Type: Object
Access: read/write
Requirement: Optional

This object contains the parent information of the function oriented reference.

`value`:

Value range: `<designation value>`
Type: String
Access: read/write
Requirement: Mandatory

This is the function oriented reference designation value of the parent function.

`local`:

Value range: `<local designation value>`
Type: String
Access: read/write
Requirement: Optional

This is the local part of the the function oriented reference of the parent function.

`oi4Identifier`:

Value range: `<oi4Identifier>`
Type: String
Access: read/write
Requirement: Optional

The `oi4Identifier` assigned to the parent function.

`product`:

Value range: `<productObject>`
Type: Object
Access: read/write
Requirement: Optional

The `product` object contains the product oriented reference. A product describes the components of which a system consists.

The `productObject` is structured as following:

`value`:

Value range: `<designation value>`
Type: String
Access: read/write
Requirement: Mandatory

This is the product oriented reference designation value.

`local`:

Value range: `<local designation value>`
Type: String
Access: read/write
Requirement: Optional

This is the local part of the the product oriented reference.

`parent`:

Value range: `<productParent>`
Type: Object
Access: read/write
Requirement: Optional

This object contains the parent information of the product oriented reference.

`value`:

Value range: `<designation value>`
Type: String
Access: read/write
Requirement: Mandatory

This is the product oriented reference designation value of the parent product.

`local`:

Value range: `<local designation value>`
Type: String
Access: read/write
Requirement: Optional

This is the local part of the the product oriented reference of the parent function.

`oi4Identifier`:

Value range: `<oi4Identifier>`
Type: String
Access: read/write
Requirement: Optional

The `oi4Identifier` assigned to the parent product.

`location`:

Value range: `<locationObject>`
Type: Object
Access: read/write
Requirement: Optional

The `location` object contains the product oriented reference. A location describes the geographical or physical position of its elements.

The `locationObject` is structured as following:

`value`:

Value range: `<designation value>`
Type: String
Access: read/write
Requirement: Mandatory

This is the location oriented reference designation value.

`local`:

Value range: `<local designation value>`
Type: String
Access: read/write
Requirement: Optional

This is the local part of the the location oriented reference.

`parent`:

Value range: `<productParent>`
Type: Object
Access: read/write
Requirement: Optional

This object contains the parent information of the location oriented reference.

`value`:

Value range: `<designation value>`
Type: String
Access: read/write
Requirement: Mandatory

This is the location oriented reference designation value of the parent location.

`local`:

Value range: `<local designation value>`
Type: String
Access: read/write
Requirement: Optional

This is the local part of the the location oriented reference of the parent location.

`oi4Identifier`:

Value range: `<oi4Identifier>`
Type: String
Access: read/write
Requirement: Optional

The `oi4Identifier` assigned to the parent location.

9.3.15 pagination

Open Industry 4.0 Alliance' payload format for any kind of data are so called OPC UA pub/sub DataSets. Each DataSet is packt in a single DataSetMessage (see 9.2.3), several of this messages can be packt for sending over the wire in a NetworkMessage (9.2.1).

Under these circumstances, a NetworkMessage may become excessively long. To prevent this, an optionally DataSetMessage with `pagination` information to indicate that there are other NetworkMessages with associated DataSetMessages is available.

Adding this `pagination` DataSet to the payload of a topic makes it possible to detect and manage subsets of huge amounts of DataSetMessages.

An example would be to request all Master Asset Models, an application has under control. Via `…/get/mam` over the Message Bus. Each `mam` would be added to a single NetworkMessage and published via `…/pub/mam` - this could damage the publisher, the subscriber and even the broker or the system, when it runs out of resources.

To avoid this, both the publisher and the subscriber can make use of the `pagination` mechanism:

- A publisher who has too many DataSetMessages for a single NetworkMessage can publish a part of the DataSetMessages and add a `pagination` object, which tells all subscribers *"This message contains 10 out of 100 DataSetMessages - we are on Page 5 now"*. The subscriber now knows, that five other publications will follow, till all related information is published.
- A subscriber who wants to know something and asks therefor with …/`get`/`<resource>` can add a `pagination` object to its payload to define, how much DataSetMessages the NetworkMessage may have as a maximum, so the subscriber is able to handle it.

For standard publications, which were not triggered through a previously received get message, the publisher defines the maximum size by its own. If this messages are too long for a specific application, this application must re-trigger publishing the information with use of `pagination`.

In general, `pagination` information can be used in any context, for every `method` and in combination with every `resources`.

The complete `pagination` object has the same structure for the methods `pub`, `set`, `get` and `del`, however, they differ in which properties are mandatory. The differences are explained in 9.3.15.1 for `get` and in 9.3.15.2 for `pub`, `set` and `del`.

An example of its use is explained in appendix B1.15.

9.3.15.1 pagination (get)

The payload for the method `get` of a DataSetMessage (9.2.3) containing `pagination` information is structured as followed:

`totalCount`:

Value range: `<number>`
Type: UInt32
Access: read only
Requirement: Optional

A published message to the topic methods `pub`, `set` or `del` might use `totalCount` to announce, how many of the actual processed resources can be expected.
The amount of included DataSetMessages out of `totalCount` can be calculated from existing NetworkMesage or is given by the key `perPage`.

NOTE *There is no use of* `totalCount` *in context of the method* `get`*.*

`perPage`:

Value range: `<number>`
Type: UInt32
Access: read only
Requirement: Mandatory

A published message to the topic methods `pub`, `set` or `del` might use `perPage` to announce, how many of the `totalCount` DataSetMessages can be included in this publication.
A published message to the topic method `get` might use `perPage` to announce, how many DataSetMessages it expects maximal in the related publication.

`page`:

Value range: `<number>`
Type: UInt32
Access: read only
Requirement: Mandatory

If several NetworkMessages has to be published to transport the relevant DataSetMessages, the value of `page` shows which page is published or requested.

`hasNext`:

Value range: `<true or false>`
Type: Boolean
Access: read only
Requirement: Optional

`hasNext` shows if an additional NetworkMessage can be expected.

NOTE *There is no use of* `hasNext` *in context of the method* `get`*.*

An example of its use is explained in appendix B1.1.15.

9.3.15.2 pagination (pub, set, del)

The payload for the methods `pub`, `set` and `del` of a DataSetMessage (9.2.3) containing `pagination` information is structured as followed:

`totalCount`:

Value range: `<number>`
Type: UInt32
Access: read only
Requirement: Mandatory

A published message to the topic methods `pub`, `set` or `del` might use `totalCount` to announce, how many of the actual processed resources can be expected.
The amount of included DataSetMessages out of `totalCount` can be calculated from existing NetworkMesage or is given by the key `perPage`.

`perPage`:

Value range: `<number>`
Type: UInt32
Access: read only
Requirement: Mandatory

A published message to the topic methods `pub`, `set` or `del` might use `perPage` to announce, how many of the `totalCount` DataSetMessages can be included in this publication.
A published message to the topic method `get` might use `perPage` to announce, how many DataSetMessages it expects maximal in the related publication.

`page`:

Value range: `<number>`
Type: UInt32
Access: read only
Requirement: Mandatory

If several NetworkMessages has to be published to transport the relevant DataSetMessages, the value of `page` shows which page is published or requested.

`hasNext`:

Value range: `<true or false>`
Type: Boolean
Access: read only
Requirement: Mandatory

`hasNext` shows if an additional NetworkMessage can be expected.

An example of its use is explained in appendix B1.1.15.

9.3.16 locale

Adding this `locale` DataSet to the payload of a topic with the method `get`, it is possible to trigger a localized `pub` message if supported by the used application.

An example would be to request the Master Asset Model, via `…/get/mam/<oi4Identifier>`. Each `mam` contains LocalizedText information, which are localized to English by default. If an application supports localization and the requested language is implemented to the application, the following `pub` message, containing the `CorrelationId` of the `get` message and will provide all LocalizedText elements in the requested language.

The DataSetMessage with `locale` information can be added to all methods but its obvious meaning is related to `get`. The `get` method combined with `local` payload, triggers a publication in a specific language.

The payload of a DataSetMessage (9.2.3) containing `locale` information is structured as followed:

`locale`:

Value range: `<LocaleId>`
Type: String
Access: read only
Requirement: Optional

The `localeId`, which looks linke `"en-US"` or `"de-DE"` (9.2) will be placed in `locale` as String.

NOTE *The information is provided via the* `<method>` `get` *and can be combined with any* `resource`*, which might have LocalizedText elements in it.*

An example of its use is explained in appendix B1.1.16.

9.4 OPC UA methods, defined by the Alliance

This chapter explains the most necessary methods and related data types, which the Alliance are using for call/reply pattern over the Message Bus.

All the objects are JSON encoded.

9.4.1 fileUpload

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The input and output arguments, described in the following, are used by the method `fileUpload`, which is mentioned as common service in 8.1.5.2 and used in 10.2.1.

Because `fileUpload` is a common service, globally defined by the Alliance, a DataSetClassId, used by the ServiceNetworkMessage (9.2.16), is available for this service (A2).

9.4.2 fileDownload

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The input and output arguments, described in the following, are used by the method `fileDownload`, which is mentioned as common service in 8.1.5.2 and used in 10.2.2.

Because `fileDownload` is a common service, globally defined by the Alliance, a DataSetClassId, used by the ServiceNetworkMessage (see 9.2.16), is available for this service (see A2).

9.4.3 firmwareUpdate

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The input and output arguments, described in the following, are used by the method `firmwareUpdate`, which is mentioned as common service in 8.1.5.2 and used in 10.2.3.

Because `firmwareUpdate` a common service, globally defined by the Alliance, a DataSetClassId, used by the ServiceNetworkMessage (see 9.2.16), is available for this service (see A2).

9.4.4 blink

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The input and output arguments, described in the following, are used by the method `blink`, which is mentioned as common service in 8.1.5.2 and used in 10.2.4.

Because `blink` is a common service, globally defined by the Alliance, a DataSetClassId, used by the ServiceNetworkMessage (see 9.2.16), is available for this service (see A2).

9.4.5 newDataSetWriterId

In some cases it is required to set a new DataSetMessage in an Application from outside the application. E.g. a reference designation (9.3.14), known by a MES system shall be written into an OTConnector. Therefore the ITConnector, dealing with the MES system, has to set/create the referenceDesignation inside the OTConnector. To do so, a valid and unique DataSetWriterId for the OTConnector is necessary.

The service `newDataSetWriterId` requests a new, so far unused, DataSetWriterId from the Publisher, which consumes this method call.

The input and output arguments, described in the following, are used by the method `newDataSetWriterId`, which is mentioned as common service in 8.1.5.2 and used in 10.2.5.

Because `newDataSetWriterId` is a common service, globally defined by the Alliance, a DataSetClassId, used by the ServiceNetworkMessage (9.2.16), is available for this service (A2).

The serviceParameterRequest (9.2.17) contains objects of type methodsToCall, which are an array of CallMethodRequest (9.2.18).

The `inputArguments` of the CallMethodRequest object are:

`resource`:

Value range: `<resourceType>`
Type: String
Access: read/write
Requirement: Mandatory

`resource` (8.1.5.1) contains the resource the requested DataSetWriterId shall be used for.

The serviceParameterResponse (9.2.19) contains objects of type results, which are an array of CallMethodResult (9.2.20).

The `outputArguments` of the CallMethodResult are:

`DataSetWriterId`:

Value range: `<UINT16>`
Type: UInt16
Access: read/write
Requirement: Mandatory

`DataSetWriterId` contains the DataSetWriterId, which is provided from the Publisher to use for the upcoming set request.

`ttl`:

Value range: `<UINT32 in seconds>`
Type: UInt32
Access: read/write
Requirement: Mandatory

`ttl` contains a value, which describes the time to live of the provided

`DataSetWriterId`. Initial usage of the provided `DataSetWriterId` is only possible during this period of time.

10 Message Bus communication

In chapter 8 the topic structure was described and in chapter 9 the data formats of the payload. This chapter addresses the use of both in combination.

Because Open Industry 4.0 Alliance uses its Message Bus in a standard pub/sub pattern (10.1) and in a service-oriented call/reply pattern (10.2 and 10.3), the following provided explanations for resources and services are split into several sub-chapters.

10.1 Resources

The usage of all resources, defined in Open Industry 4.0 Alliance context (8.1.5), are described in the following sub-chapters. Resources are used from applications in a standard pub/sub pattern.

For a better understanding, on how to handle the resources in combination with the four existing methods `pub`, `get`, `set` and `del`, the four base interactions are shown in a general but simplified way in figure 13:

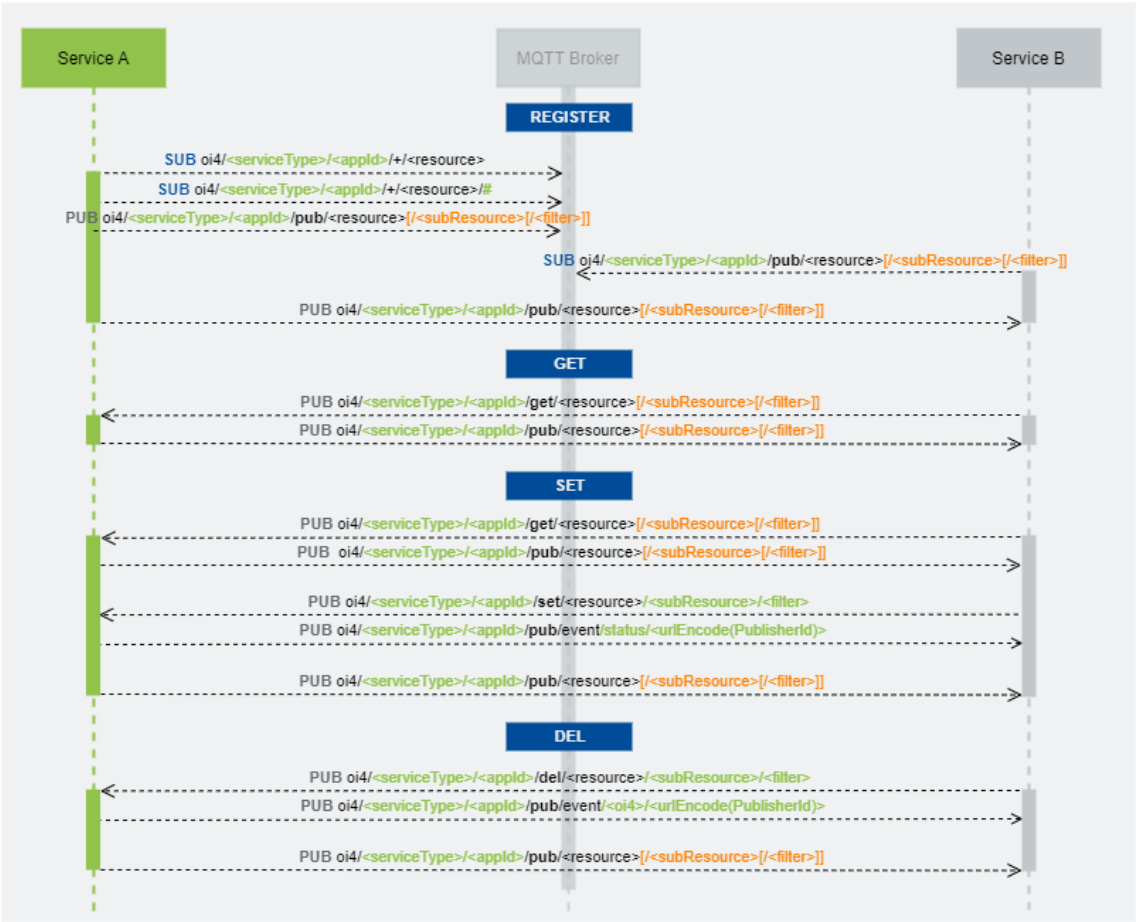

Figure 16 Sequence diagram of the four base interactions on how to use resources over the Message Bus

At startup *Service A* subscribes itself to the topics on which its reaction is expected. This typically includes `get`, `set`, and `del` methods, depending on the resource in the subscribed topic. Once done subscribing to the topics *Service A* may start publishing. As long as no other service is subscribed to the topics *Service A* publishes on, the messages just end on the

Message Bus. When *Service B* starts subscribing to topics on which *Service A* is publishing the messages from *Service A* traverse through the Message Bus and arrive at *Service B*.

In case *Service B* want to request a specific resource from *Service A*, *Service B* publishes on the `get` topic of one of *Service A*'s resources, optionally providing a `filter`. *Service A* receives the `get` message from *Service B* through the Message Bus and replies with a `pub` message over the Message Bus.

Changing a resource's content likely requires *Service B* to request the resource object from *Service A's* `get` *topic first*. *Service B* publishes the updated resource object on *Service A*'s `set` topic for this resource. Now, the `filter` is required in the topic. Service A will publish an event over the Message Bus to share the status of `set` command. In case, the `set` was sucessfull *Service A* must publish the changed resource on the `pub` topic regarding to the settings in publicationList.

To delete a resource object from *Service A*, *Service B* needs to know the `filter` of the resource it aims to delete. *Service B* then publishes on the `del` topic of *Service A*. Service A will publish an event over the Message Bus to share the status of `del` command. If the whole `filter` has been deleted *Service A* won't publish anything on the resource's topic. In case, the `del` has deleted only parts of a `filter` (e.g. a single subscription out of the subscriptionList), this is a change to *Service A* and *Service A* must publish the changed resource on the `pub` topic regarding to the settings in publicationList.

The payload and the topic are related via subResource and filter of the DataSetMessage, contained in the payload, and `subResource` and `filter` contained in the topic.

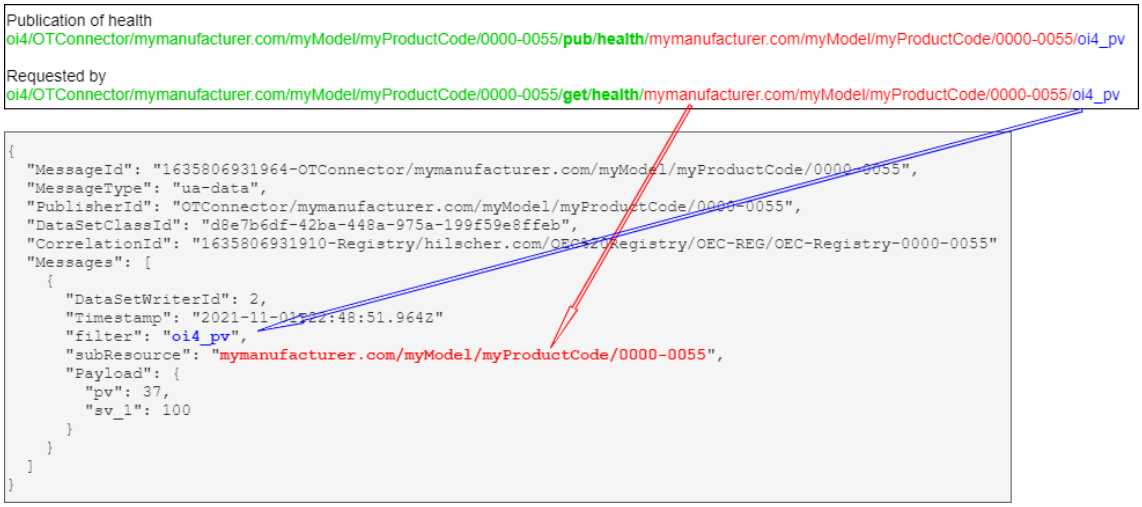

Figure 17 Relationship between topic and payload in standard pub-sub-pattern

The following sub-chapters will explain how to use the available resources in combination with the related methods.

By using the methods `set` and `get`, the application should react use case specific and, if necessary, initiate positive/negative events and/or syslog entries.

Asking for the `metadata` of `<resources>` is only possible for `data`, but not for Alliance's predefined resources such as `mam`, `health`, `config` and so on.

The metadata for all other resources than `data` must be specified by Open Industry 4.0 Alliance. The Alliance must provide the definitions and the related `DataSetClassId` for them.

All Open Industry 4.0 Alliance defined elements and objects are named in lowerCamelCase. In case Open Industry 4.0 Alliance extends an existing OPC UA object, Open Industry 4.0 Alliance follows the writing rules of the other elements inside the existing object.

10.1.1 mam (Master Asset Model)

A Master Asset Model (`mam`) represents the nameplate of a single asset (device, application, ...). It is derived from IVendorNameplateType, described in OPC UA Part 100 (Part 100-4.5.2)

The payload of a DataSetMessage (9.2.3) containing a Master Asset Model is described in detail in section 4.

For the resource `mam`, the methods `pub` and `get` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `mam` object is not included. The `subResource` can be used to request only `mam` information of a specific asset. Without `subResource`, all `mam` information will be requested.
- In combination with the method `pub`, the payload contains DatsSetMessages of type `mam` and optional `pagination` (9.3.15).

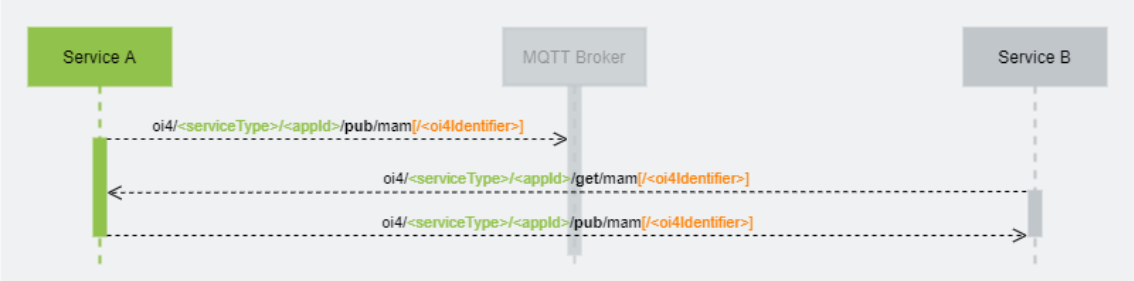

Figure 18 Sequence diagram of the two basic interactions that can be used for the resource mam

Get mam from assets

Via the `<method>/<resource>[/<subResource>]` combination `get/mam[/<oi4Identifier>]` it is possible to trigger a (re-)publishing of this information to the `pub/mam[/<oi4Identifier>]` topic.

Publisher topic:

Get explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/get/mam/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/mam`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/mam` *triggers a publication of all*

available `mam` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/mam`.

Publish mam from assets

The information is provided via the `<method>/<resource>[/<subResource>]` combination `pub/mam[/<oi4Identifier>]`.

Any application can listen to any new `mam` information by subscribing to the topics `oi4/+/+/+/+/+/pub/mam` and `oi4/+/+/+/+/+/pub/mam/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/mam/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/mam`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": ""
        },
        "ManufacturerUri": "",
        "Model": {
          "locale": "en-US",
          "text": ""
        },
        "ProductCode": "",
        "HardwareRevision": "",
        "SoftwareRevision": "",
        "DeviceRevision": "",
        "DeviceManual": "",
        "DeviceClass": "",
        "SerialNumber": "",
        "ProductInstanceUri": "",
        "RevisionCounter": <INT32>,
        "Description": {
          "locale": "en-US",
          "text": ""
        }
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

NOTE *More advance examples can be found in the appendix B1.1.1.*

10.1.2 health

The resource `health` represents the actual condition of a single asset (device, application, …). It is derived from IDeviceHealthType, described in OPC UA Part 100 (Part 100-4.5.4), but extended by an additional key called `healthScore`.

The payload of a DataSetMessage (9.2.3) containing `health` information is described in detail in section 9.3.2.

For the resource `health`, the methods `pub` and `get` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination`

(9.3.15) and `locale` (9.3.16). A `health` object is not included. The `subResource` can be used to request only `helth` information of a specific asset. Without `subResource`, all `health` information will be requested.

- In combination with the method `pub`, the payload contains DataSetMessages of type `health` and optional `pagination` (9.3.15).

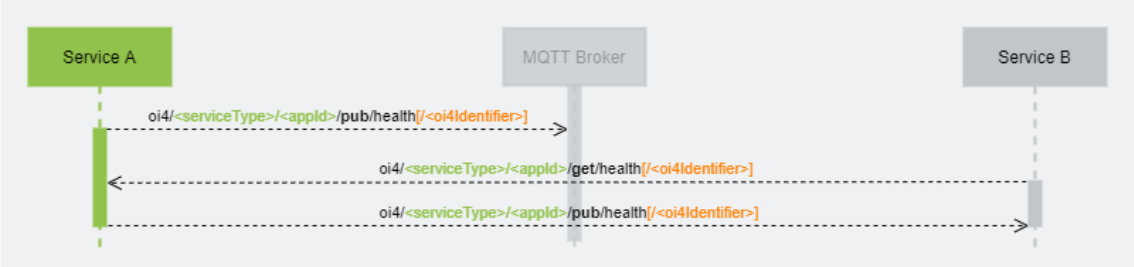

Figure 19 Sequence diagram of the two basic interactions that can be used for the resource health

Get health from assets

Via the `<method>/<resource>[/<subResource>]` combination `get/health[/<oi4Identifier>]`, the publication of this information to the `pub/health[/<oi4Identifier>]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/get/health/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/health`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/health` *triggers a publication of all available* `health` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/health`.

Publish health from assets

The information is provided via the `<method>/<resource>[/<subResource>]` combination `pub/health[/<oi4Identifier>]`.

Any application can listen to any new `health` information by subscribing to the topics `oi4/+/+/+/+/+/pub/health` and `oi4/+/+/+/+/+/pub/health/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/health/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/health`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "health": "<<enumName>_<enumValue>>",
        "healthScore": <Byte>
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic as specified in the pagination object, till all DataSetMessages are published.*

NOTE *More advance examples can be found in the appendix B1.1.2.*

10.1.3 config

The `config` resource represents the actual configuration of an asset (device, application, …). Several configuration DataSetMessages may exist for an asset.

The payload of a DataSetMessage (9.2.3) containing `config` information is described in detail in section 9.3.3.

For the resource `config`, the methods `pub`, `get` and `set` are available. As the device or application providing the resource is the owner of the configuration object, it is the only instance which can remove objects from within the config object. Therefore the `del` is not supported. However, to “delete” or better unset a configured object requires sending a value with an empty string with the `set` method.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `config` object is not included. The `subResource` can

be used to request only `config` objects of a specific asset. The `filter` can be used to request a specific `config` object. Without `subResource` and `filter`, all `config` objects will be requested.

- In combination with the method `pub`, the payload contains DataSetMessages of type `config` and optional `pagination` (9.3.15).
- In combination with the method `set`, the payload contains a single DataSetMessages of type `config`. A topic with method `set` must contain `resource`, `subResource` and `filter` to be valid.

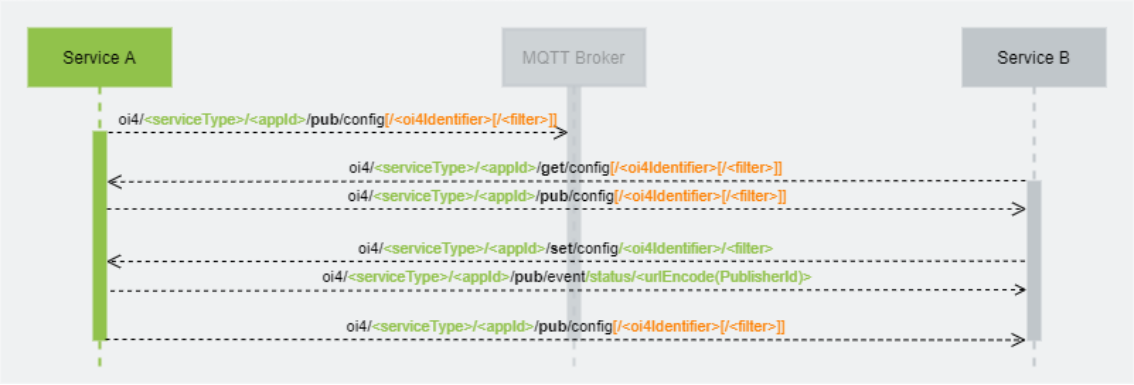

Figure 20 Sequence diagram of the three basic interactions that can be used for the resource config

Get config from assets

Via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `get/config[/<oi4Identifier>[/<filter>]]` it is possible to trigger a (re-)publishing of this information to the `pub/config[/<oi4Identifier>[/<filter>]]` topic.

Publisher topic:

Get explicit DataSetMessage for a specific `filter`:
`oi4/<serviceType>/<appId>/get/config/<oi4Identifier>/<filter>`

Get all DataSetMessages for a specific asset:
`oi4/<serviceType>/<appId>/get/config/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/config`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/config` *triggers a publication of*

all available `config` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/config`.

Publish config from assets

The information is provided via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `pub/config[/<oi4Identifier>[/<filter>]]`. Any application can listen to any new `config` information by subscribing to the topics `oi4/+/+/+/+/+/pub/config` and `oi4/+/+/+/+/+/pub/config/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific `filter`:
`oi4/<serviceType>/<appId>/pub/config/<oi4Identifier>/<filter>`

Publish all DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/config/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/config`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<object name;
                 might correlate with content of context object>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "<groupName>": {
          "name": {
            "locale": "en-US", "text": "<name>"
          },
          "description": {
            "locale": "en-US", "text": "<description>"
          },
          "<configNameA>": {
            "description": {
              "locale": "en-US", "text": "<description>"
            },
            "name": {
              "locale": "en-US", "text": "<name>"
            },
            "type": "<Subset of OPC UA Base Types>",
            "validation": {
              "pattern": "<string>"
            },
            "value": "<actual value>"
          },
          "<configNameB>": {
            "description": {
              "locale": "en-US", "text": "<description>"
            },
            "name": {
              "locale": "en-US", "text": "<name>"
            },
            "type": "<Subset of OPC UA Base Types>",
            "validation": {
              "min": <REAL>,
              "max": <REAL>
            },
            "value": "<actual value>"
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "<DisplayName of whole configuration object>"
          },
          "description": {
            "locale": "en-US",
            "text": "<Description of whole configuration object>"
          }
        }
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

Set config from specific asset (set new values)

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subResource>/<filter>` combination `set/config/<oi4Identifier>/<filter>` it is possible to modify configurations. In a `set` message, all config elements which are mandatory `("mandatory": true)` must be provided! After a `set` message arrived, the application will publish an `event` message with status information about the `set` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `set` *must be done explicit. That means* `set/config` *is only possible by using a complete topic containing* `resource`*,* `subresource` *and* `filter`*, such as* `oi4/<serviceType>/<appId>/set/config/<oi4Identifier>/<filter>`*. Setting multiple DataSetMessages at once via* `oi4/<serviceType>/<appId>/set/config` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

If setting `config` was successful, the new `config` will be published to the `pub/config[/<subResource>[/<filter>]]` topic, according the the settings in publicationList (9.3.11).

Publisher topic:

Set explicit DataSetMessage for a specific `filter`:
`oi4/<serviceType>/<appId>/get/config/<oi4Identifier>/<filter>`

Payload:

The payload must contain the values of mandatory config names. In addition, it may also contain the other properties received through `pub/config`. For details on the `set` format see section 9.3.3.2

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<object name; might
                 correlate with content of context object>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
          "<configNameA>": {
            "value": "<new value>"
          },
          "<configNameB>": {
            "value": "<new value>"
          }
        }
      }
    }
  ]
}
```

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<object name; might correlate with content of context
object>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "<groupName>": {
          "name": {
            "locale": "en-US", "text": "<name>"
          },
          "description": {
            "locale": "en-US", "text": "<description>"
          },
          "<configNameA>": {
            "description": {
              "locale": "en-US", "text": "<description>"
            },
            "name": {
              "locale": "en-US", "text": "<name>"
            },
            "type": "<Subset of OPC UA Base Types>",
            "validation": {
              "pattern": "<string>"
            },
            "value": "<new value>"
          },
          "<configNameB>": {
            "description": {
              "locale": "en-US", "text": "<description>"
            },
            "name": {
              "locale": "en-US", "text": "<name>"
            },
            "type": "<Subset of OPC UA Base Types>",
            "validation": {
              "min": <REAL>,
              "max": <REAL>
            },
            "value": "<new value>"
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "<DisplayName of whole configuration object>"
          },
          "description": {
            "locale": "en-US",
            "text": "<Description of whole configuration object>"
          }
        }
      }
    }
  ]
}
```

Set config from specific asset (empty existing values)

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

While actual deletion of configuration is not possible from outside the service, providing the configuration parameters, it is possible to undo a configured value. To do so simply requires a `set` with empty value of the config object.

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<object name;
                 might correlate with content of context object>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
          "<configNameA>": {
            "value": ""
          },
          "<configNameB>": {
            "value": ""
          }
        }
      }
    }
  ]
}
```

NOTE *More advance examples can be found in the appendix B1.1.3.*

10.1.4 license

The resource `license` represents the actual license information for a single application.

The payload of a DataSetMessage (9.2.3) containing `license` information is described in detail in section 9.3.4.

For the resource `license`, the methods `pub` and `get` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `license` object is not included. The `subResource` can be used to request only `license` information of a specific asset. Without `subResource`, all `license` information will be requested. Additional to the `subResource`, a `filter` can be set to request a specific `licenseId` of a specific asset.
- In combination with the method `pub`, the payload contains DataSetMessages of type `license` and optional `pagination` (9.3.15).

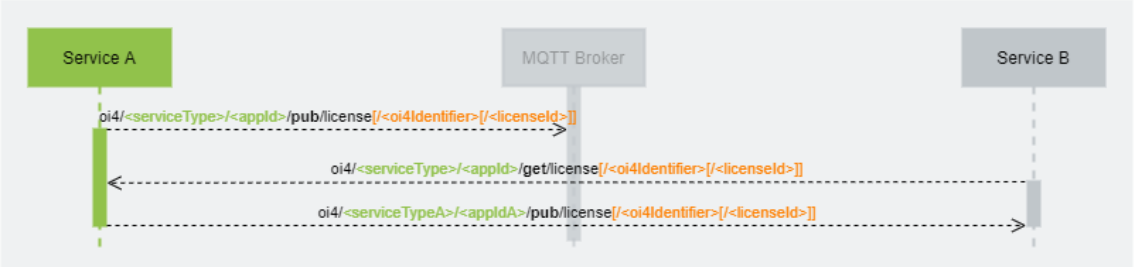

Figure 21 Sequence diagram of the two basic interactions that can be used for the resource license

Get license from assets

Via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `get/license[/<oi4Identifier>[/<licenseId>]]`, the publication of this information to the `pub/license[/<oi4Identifier>[/<licenseId>]]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific `licenseId`:
`oi4/<serviceType>/<appId>/get/license/<oi4Identifier>/<licenseId>`

Get all DataSetMessages for a specific asset:
`oi4/<serviceType>/<appId>/get/license/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/license`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/license` *triggers a publication of all available* `license` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/license`.

Publish license from assets

The information is provided via
the `<method>/<resource>[/<subResource>[/<filter>]]` combination `pub/license[/<oi4Identifier>[/<licenseId>]]`.

Any application can listen to any new `license` information by subscribing to the topics `oi4/+/+/+/+/+/pub/license` and `oi4/+/+/+/+/+/pub/license/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific `licenseId`:
`oi4/<serviceType>/<appId>/pub/license/<oi4Identifier>/<licenseId>`

Publish all DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/license/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/license`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "Apache%202.0",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "components": [
          {
            "component": "MQTTCl",
            "licAuthors": [
              "Lorem ipsum"
            ],
            "licAddText": "additional to standard text"
          },
          {
            "component": "nodeOPC",
            "licAuthors": [
              "Lorem ipsum"
            ],
            "licAddText": ""
          }
        ]
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

NOTE *More advance examples can be found in the appendix B1.1.4.*

10.1.5 licenseText

The resource `licenseText` represents the actual license text for a specific `licenseId` information for a single application.

The payload of a DataSetMessage (9.2.3) containing `licenseText` information is described in detail in section 9.3.5.

For the resource `licenseText`, the methods `pub` and `get` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `licenseText` object is not included. The `subResource` can be used to request only `licenseText` information of a specific asset. Without `subResource`, all `licenseText` information will be requested. Additional to the `subResource`, a `filter` can be set to request the `licenseText` to a specific `licenseId` of a specific asset.
- In combination with the method `pub`, the payload contains DatsSetMessages of type `licenseText` and optional `pagination` (9.3.15).

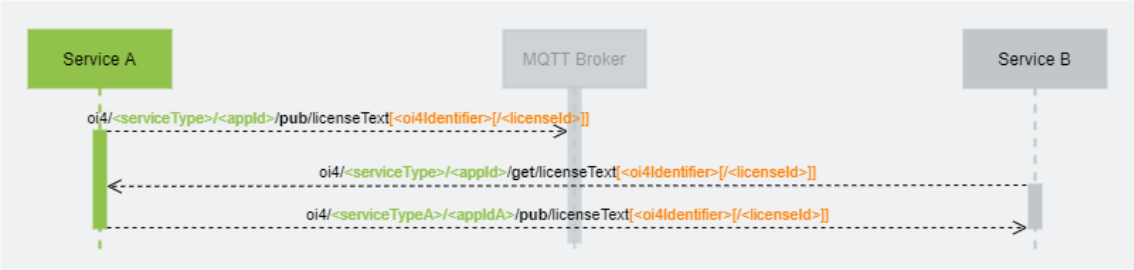

Figure 22 Sequence diagram of the two basic interactions that can be used for the resource licenseText

Get licenseText from assets

Via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `get/licenseText[/<oi4Identifier>[/<licenseId>]]`, the publication of this information to the `pub/licenseText[/<oi4Identifier>[/<licenseId>]]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific `licenseId`:
`oi4/<serviceType>/<appId>/get/licenseText/<oi4Identifier>/<licenseId>`

Get all DataSetMessages for a specific asset:
`oi4/<serviceType>/<appId>/get/licenseText/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/licenseText`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/licenseText` *triggers a publication of all available* `licenseText` information*, each in its own DataSetMessage added*

to a NetworkMessage and published to the topic `oi4/<serviceType>/<appId>/pub/licenseText`.

Publish licenseText from assets

The information is provided via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `pub/licenseText[/<oi4Identifier>[/<licenseId>]]`.

Any application can listen to any new `licenseText` information by subscribing to the topics `oi4/+/+/+/+/+/pub/licenseText` and `oi4/+/+/+/+/+/pub/licenseText/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific `licenseId`:
`oi4/<serviceType>/<appId>/pub/licenseText/<oi4Identifier>/<licenseId>`

Publish all DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/licenseText/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/licenseText`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<licenseId>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "licenseText": ""
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic as specified in the pagination object, till all DataSetMessages are published.*

NOTE *More advance examples can be found in the appendix B1.1.5.*

10.1.6 rtLicense

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The resource `rtLicense` represents the actual runtime license(s) information for a single application.

The payload of a DataSetMessage (9.2.3) containing `rtLicense` information is described in detail in section 9.3.6.

For the resource `rtLicense`, the methods `pub` and `get` are available.

- n combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `rtLicense` object is not included. The `subResource` can be used to request only `rtLicense` information of a specific asset. Without `subResource`, all `rtLicense` information will be requested.
- In combination with the method `pub`, the payload contains DatsSetMessages of type `rtLicense` and optional `pagination` (9.3.15).

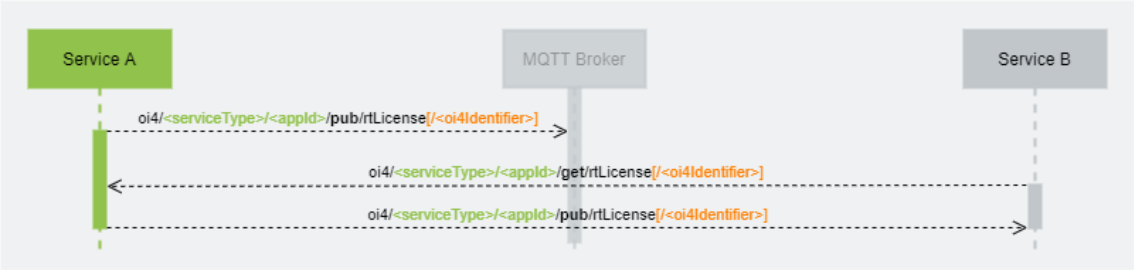

Figure 23 Sequence diagram of the two basic interactions that can be used for the resource rtLicense

Get rtLicense from assets

Via the `<method>/<resource>[/<subResource>]` combination `get/rtLicense[/<oi4Identifier>]`, the publication of this information to the `pub/rtLicense[/<oi4Identifier>]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/get/rtLicense/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/rtLicense`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "ebd12d4b-da1c-4671-ab86-db102fecc603",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/rtLicense` *triggers a publication of all available* `rtLicense` *information, each in its own DataSetMessage added to a*

NetworkMessage and published to the topic `oi4/<serviceType>/<appId>/pub/rtLicense`.

Publish rtLicense from assets

The information is provided via the `<method>/<resource>[/<subResource>]` combination `pub/rtLicense[/<oi4Identifier>]`.

Any application can listen to any new `rtLicense` information by subscribing to the topics `oi4/+/+/+/+/+/pub/rtLicense` and `oi4/+/+/+/+/+/pub/rtLicense/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/rtLicense/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/rtLicense`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "ebd12d4b-da1c-4671-ab86-db102fecc603",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "",
      "subResource": "<oi4Identifier>",
      "Timestamp": "DataType",
      "Payload": {
        <T O D O>
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

NOTE *More advance examples can be found in the appendix B1.1.6.*

10.1.7 data

Applications can offer DataSets "on their own", and pack them into a DataSetMessage, which is then available via `data`.

The payload of a DataSetMessage (9.2.3) containing `data` is not defined by the Alliance. Each data object might look different, but it has to follow the rules of OPC Foundation's specification for JSON coded pub/sub DataSets (OPC UA Part 14-7.2.3.3-Table 92).

NOTE *In addition to the application driven DataSets, the Alliance defines an optional DataSet to*

access the available process values in a standard way (9.3.7). To reach this standardized DataSet the filter `oi4_pv` is used for every asset supporting this feature.

For the resource `data`, the methods `pub`, `get` and `set` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `data` object is not included. The `subResource` can be used to request only `data` objects of a specific asset. The `filter` can be used to request a specific `data`. Without `subResource` and `filter`, all `data` objects will be requested.
- In combination with the method `pub`, the payload contains DataSetMessages of type `data` and optional `pagination` (9.3.15).
- In combination with the method `set`, the payload contains a single DataSetMessages of type `data`. A topic with method `set` must contain `resource`, `subResource` and `filter` to be valid.

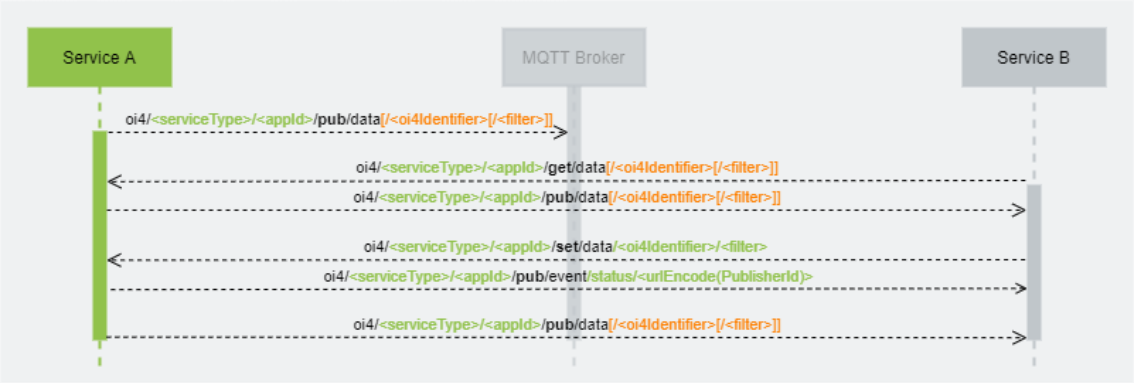

Figure 24 Sequence diagram of the three basic interactions that can be used for the resource data

Get data from assets

Via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `get/data[/<oi4Identifier>[/<filter>]]`, the publication of this information to the `pub/data[/<oi4Identifier>[/<filter>]]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific `filter`:
`oi4/<serviceType>/<appId>/get/data/<oi4Identifier>/<filter>`

Get all DataSetMessages for a specific asset:
`oi4/<serviceType>/<appId>/get/data/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/data`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/data` *triggers a publication of all available* `data` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/data`.

Publish data from assets

The information is provided via
the `<method>/<resource>[/<subResource>[/<filter>]]` combination
`pub/data[/<oi4Identifier>[/<filter>]]`.

Any application can listen to any new `data` information by subscribing to the topics `oi4/+/+/+/+/+/pub/data` and `oi4/+/+/+/+/+/pub/data/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific `filter`:
`oi4/<serviceType>/<appId>/pub/data/<oi4Identifier>/<filter>`

Publish all DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/data/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/data`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload for asset defined `filter`:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<filter>",
      "subResource": "<oi4Identifier>",
      "SequenceNumber": <UINT32>,
      "MetaDataVersion": {
        "majorVersion": <UINT32>,
        "minorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Status": <UINT32>,
      "Payload": {
        <DataSet>
      }
    }
  ]
}
```

Payload if `filter` equals `oi4_pv`:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "oi4_pv",
      "subResource": "<oi4Identifier>",
      "SequenceNumber": <UINT32>,
      "MetaDataVersion": {
        "majorVersion": <UINT32>,
        "minorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Status": <UINT32>,
      "Payload": {
        "pv": <e.g. subset of OPC UA Base Types>,
        "sv_1": {<e.g. complex JSON object>},
        "sv_2": [<e.g. array>],
        "sv_<n>": {}
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

Set data from specific asset

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subRecource>/<filter>` combination `set/data/<oi4Identifier>/<filter>` it is possible to modify DataSets. After a `set` message arrived, the application will publish an `event` message with status information about the `set` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `set` *must be done explicit. That means* `set/data` *is only possible by using a complete topic containing* `resource`, `subresource` *and* `filter`, *such as* `oi4/<serviceType>/<appId>/set/data/<oi4Identifier>/<filter>`. *Setting multiple DataSetMessages at once via* `oi4/<serviceType>/<appId>/set/data` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

If setting `data` was successful, the new `data` will be published to the `pub/data[/<subResource>[/<filter>]]` topic, according the the settings in publicationList (9.3.11).

Publisher topic:

Set explicit DataSetMessage for a specific `filter`:
`oi4/<serviceType>/<appId>/set/data/<oi4Identifier>/<filter>`

Payload for asset defined `filter`:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId_of_DataSetOwner>",
  "DataSetClassId": "<(meta)data specific GUID>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<filter>",
      "subResource": "<oi4Identifier>",
      "MetaDataVersion": {
        "majorVersion": <UINT32>,
        "minorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Payload": {
        <DataSet>
      }
    }
  ]
}
```

Payload if `filter` equals `oi4_pv`:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId_of_DataSetOwner>",
  "DataSetClassId": "<(meta)data specific GUID>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "oi4_pv",
      "subResource": "<oi4Identifier>",
      "MetaDataVersion": {
        "majorVersion": <UINT32>,
        "minorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Payload": {
        "pv": <e.g. subset of OPC UA Base Types>,
        "sv_1": {<e.g. complex JSON object>},
        "sv_2": [<e.g. array>],
        "sv_<n>": {}
      }
    }
  ]
}
```

NOTE *To be able to interpret all aspects of* `data`*, it might be necessary to read the* `metadata` *(see 10.1.8), corresponding to the* `data`*.*

NOTE *More advance examples can be found in the appendix B1.1.7.*

10.1.8 metadata

Assets can offer `data`, `health`, `license`, `mam` and other resources, which are accessible. To interpret this information it might be necessary to get additional information through `metadata`.

NOTE *Currently only* `data` *is described via* `metadata` *- not* `health`*,* `license` *and all the other resources. The alliance offer defined* `DataSetClassIds` *with predefined metadata in appendix A2 for resources other than* `data`*.*

For the resource `metadata`, the methods `pub`, `get` and `set` are available.

- In combination with the method `get`, the payload requires an empty DataSetMetaData. The topic must contain `resource`, `subResource` and `filter` to be valid.
- In combination with the method `pub`, the payload contains a filled DataSetMetaData.
- In combination with the method `set`, the payload contains a filled DataSetMetaData. The topic must contain `resource`, `subResource` and `filter` to be valid.

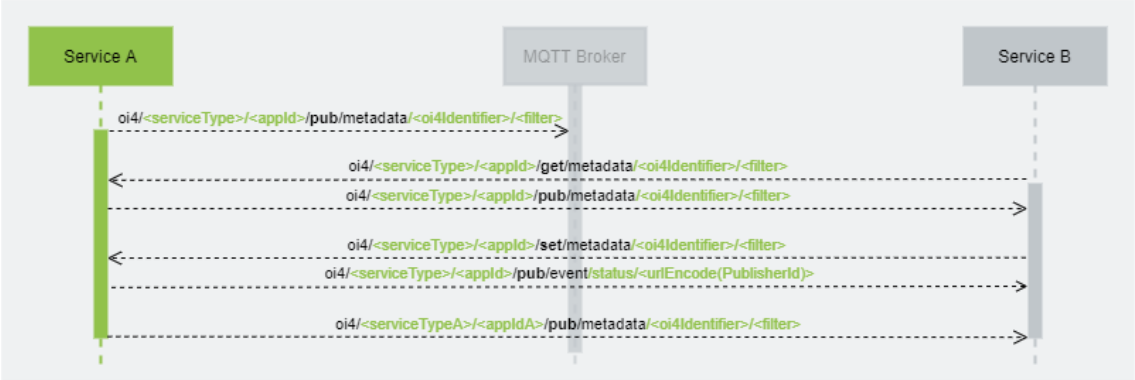

Figure 25 Sequence diagram of the three basic interactions that can be used for the resource metadata

Get metadata from specific asset/filter

Via the `<method>/<resource>/<subResource>/<filter>` combination `get/metadata/<oi4Identifier>/<filter>`, the publication of this information to the `pub/metadata/<oi4Identifier>/<filter>` topic can be triggered.

Publisher topic:

Get explicit DataSetMetaData for a specific `filter`:
`oi4/<serviceType>/<appId>/get/metadata/<oi4Identifier>/<filter>`

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-metadata",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetWriterId": <UINT16>,
  "filter": "<filter>",
  "subResource": "<oi4Identifier>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "MetaData": {}
}
```

Publish metadata from specific asset/filter

The information is provided via the `<method>/<resource>/<subResource>/<filter>` combination `pub/metadata/<oi4Identifier>/<filter>`.

Any application can listen to any `metadata` by subscribing to the topic `oi4/+/+/+/+/+/pub/metadata/#`.

Publisher topic:

Publish explicit DataSetMetaData for a specific `filter`:
`oi4/<serviceType>/<appId>/pub/metadata/<oi4Identifier>/<filter>`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-metadata",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetWriterId": <UINT16>,
  "filter": "<filter>",
  "subResource": "<oi4Identifier>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "MetaData": {
    <DataSetMetaDataType>
  }
}
```

Set metadata from specific asset/filter

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subResource>/<filter>` combination `set/metadata/<oi4Identifier>/<filter>` it is possible to modify the `metadata`. After a `set` message arrived, the application will publish an `event` message with status information about the `set` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `set` *must be done explicit. That means* `set/metadata` *is only possible by using a complete topic containing* `resource`, `subresource` *and* `filter`, *such as* `oi4/<serviceType>/<appId>/set/metadata/<oi4Identifier>/<filter>`.

If setting `metadata` was successful, the new `metadata` will be published to the `pub/metadata/<oi4Identifier>/<filter>` topic, according the the settings in publicationList (9.3.11).

Publisher topic:

Set explicit DataSetMetaData for a specific `filter`:
`oi4/<serviceType>/<appId>/get/metadata/<oi4Identifier>/<filter>`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-metadata",
  "PublisherId": "<serviceType>/<appId_of_metadataOwner>",
  "DataSetWriterId": <UINT16>,
  "filter": "<filter>",
  "subResource": "<oi4Identifier>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "MetaData": {
    <DataSetMetaDataType>
  }
}
```

NOTE *More advance examples can be found in the appendix B1.1.8.*

10.1.9 event

The resource `event` represents notifications such as wire-break, out-of-specification warnings (e.g. NE107), or errors but also information about informing events like a sensor exchange. The event represents information from both physical devices and applications.

Event messages provide resource-specific filters that are defined by the individual sub resource. These filters can be used to limit the number of received events by subscribing to a dedicated topic.

The `subResource` `<category>` as well as the `filter` `<eventLevel>` is part of the topic and can therefore be used for filtering. A topic without `category` and `eventLevel` is not allowed.

The payload of a DataSetMessage (9.2.3) containing `event` information is described in detail in section 9.3.9.

Events are only notifications and not persisted. Therefore, exclusively the method `pub` is available.

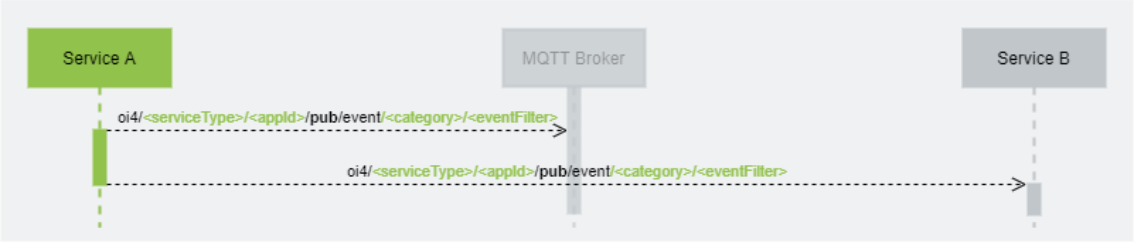

Figure 26 Sequence diagram of the basic interaction that can be used for the resource event

Publish event

The information is provided by the `<method>/<resource>/<subResource>/<filter>` combination `pub/event/<category>/<eventFilter>`.

Applications can listen to any new event by subscribing to the topic `oi4/+/+/+/+/+/pub/event/#`.

Applications can filter for all events of a specific category by subscribing to the topic `oi4/+/+/+/+/+/pub/event/<category>/#`.

Publisher topic:

Explicit DataSetMessage for a single `eventFilter`:
`oi4/<serviceType>/<appId>/pub/event/<category>/<eventFilter>`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<eventFilter>",
      "subResource": "<category>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "origin": "<oi4Identifier>",
        "number": <UINT32>,
        "description": "<string>",
        "category": "<category>"
        "details": {
          <depending on the event category>
        }
      }
    }
  ]
}
```

NOTE *Only DataSetMessages of type* `event` *with same* `category` *and* `eventFilter` *can be combined in a single NetworkMessage.*

Events are divided into the following `categories`, depending on the source:

- Syslog (see 9.3.9.2)
  - Topic `subResource` for this category is `syslog`
  - Topic `filter` for this category are described in 8.1.7/syslog
  - `category` in Payload is `CAT_SYSLOG_0`
- Status (9.3.9.1)
  - Topic `subResource` for this category is `status`
  - Topic `filter` for this category are described in 8.1.7/status
  - `category` in Payload is `CAT_STATUS_1`
- NAMUR NE107 (see 9.3.9.3)
  - Topic `subResource` for this category is `ne107`
  - Topic `filter` for this category are described in 8.1.7/ne107
  - `category` in Payload is `CAT_NE107_2`
- Generic (9.3.9.4)
  - Topic `subResource` for this category is `generic`
  - Topic `filter` for this category are described in 8.1.7/generic
  - `category` in Payload is `CAT_GENERIC_99`

NOTE *More advance examples can be found in the appendix B1.1.9.*

10.1.10 profile

The resource `profile` represents the actual available resources in a single application or device.

The `profile` contains an array of all mandatory and optional resources (8.1.5) which are supported.

The payload of a DataSetMessage (9.2.3) containing `profile` information is described in detail in section 9.3.10.

For the resource `profile`, the methods `pub` and `get` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `profile` object is not included. The `subResource` can be used to request only `profile` information of a specific asset. Without `subResource`, all `profile` information will be requested.
- In combination with the method `pub`, the payload contains DatsSetMessages of type `profile` and optional `pagination` (9.3.15).

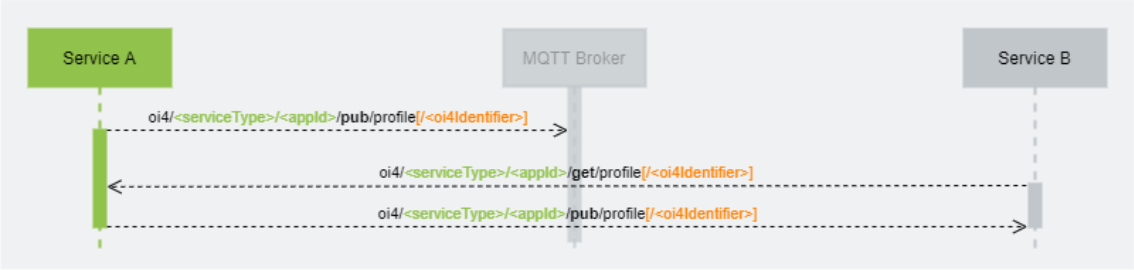

Figure 27 Sequence diagram of the two basic interactions that can be used for the resource profile

Get profile from assets

Via the `<method>/<resource>[/<subResource>]` combination `get/profile[/<oi4Identifier>]`, the publication of this information to the `pub/profile[/<oi4Identifier>]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/get/profile/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/profile`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "48017c6a-05c8-48d7-9d85-4b08bbb707f3",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/profile` *triggers a publication of all available* `profile` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/profile`.

Publish profile from assets

The information is provided via the `<method>/<resource>[/<subResource>]` combination `pub/profile[/<oi4Identifier>]`.

Any application can listen to any new `profile` information by subscribing to the topics `oi4/+/+/+/+/+/pub/profile` and `oi4/+/+/+/+/+/pub/profile/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/profile/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/profile`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "48017c6a-05c8-48d7-9d85-4b08bbb707f3",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "resource": [
          "mam", "health", "license", "licenseText", "rtLicense",
          "profile", "publicationList", "subscriptionList"
        ]
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

NOTE *More advance examples can be found in the appendix B1.1.10.*

10.1.11 publicationList

The resource `publicationList` represents the actual publications available in a single application and their rudimentary settings. This includes all publications of the application itself as well as all DataSetMessages provided by underlying devices.

The `publicationList` contains an array of DataSetMessage objects, each representing an available publication with its resource, subResource, filter, unique DataSetWriterId and several configuration settings.

Each time a DataSetMessage or a list of DataSetMessages is added/deleted/reconfigured, the `publicationList` is (re)published. If the publication is a response to a reconfiguration, only the changes along with the CorrelationId from the call that triggered the reconfiguration may be published.

NOTE *All DataSetMessages of resource* `publicationList` *are reusing the same DataSetWriterId, because each publicationList entry has the same DataSet.*

The payload of a DataSetMessage (9.2.3) containing `publicationList` information is described in detail in section 9.3.11.

For the resource `publicationList`, the methods `pub`, `get` and `set` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `publicationList` object is not included. The `subResource` can be used to request only `publicationList` objects of a specific asset. The `filter` can be used to request `publicationList` objects of a specific `resourceType`. Without `subResource` and `filter`, all `publicationList` objects will be requested.

  NOTE *Depending on the resource to filter for, the* `filter` *can be* `<resourceType>` *or* `<resourceType>/<tag>`.
  `<resourceType>` *filters are relevant for resources without additional filter definitions (e.g. mam, health and others).* `<resourceType>/<tag>` *filters are relevant for resources, which do have filter definitions such as data, config and others.*

- In combination with the method `pub`, the payload contains DatsSetMessages of type `publicationList` and optional `pagination` (9.3.15).
- In combination with the method `set`, the payload contains a single DataSetMessages of type `publicationList`. A topic with method `set` must contain `resource`, `subResource` and `filter` to be valid.

In combination with a `resourceType` (8.1.7) it is possible to get a `publicationList` filtered for a defined resource. In addition to the `resourceType` a `tag` can be added, e.g. to address a defined DataSetMessage of the resource `data`.

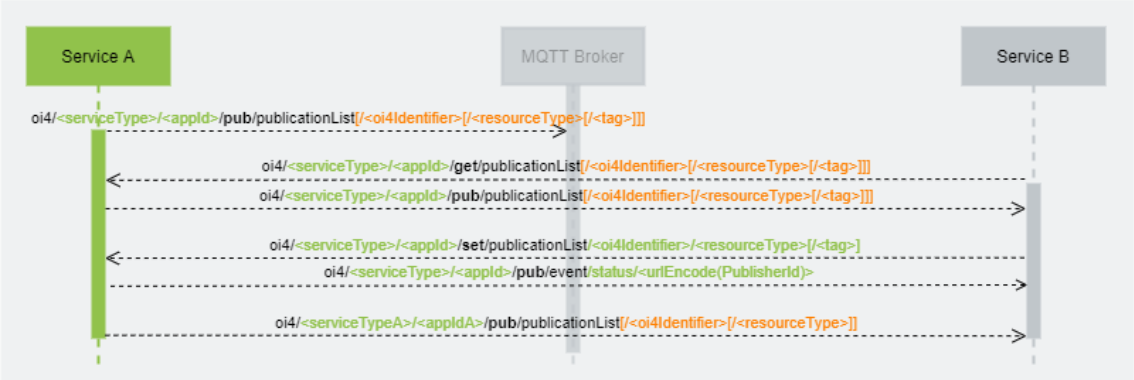

Figure 28 Sequence diagram of the three basic interactions that can be used for the resource publicationList

Get publicationList from assets

Via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `get/publicationList[/<oi4Identifier>[/<resourceType>[/<tag>]]]`, the publication of this information to the `pub/publicationList[/<oi4Identifier>[/<resourceType>[/<tag>]]]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific `resourceType[/<tag>]`:
`oi4/<serviceType>/<appId>/get/publicationList/<oi4Identifier>/<resourceType>[/<tag>]`

Get all DataSetMessages for a specific asset:
`oi4/<serviceType>/<appId>/get/publicationList/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/publicationList`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/publicationList` *triggers a publication of all available* `publicationList` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/publicationList`.

Publish publicationList from assets

The full information is provided via the `<method>/<resource>` combination `pub/publicationList`. If filtered information are wanted, the `<method>/<resource>/<subResource>[/<filter>]` combination `pub/publicationList/<oi4Identifier>[/<resourceType>[/<tag>]]` might be used.

Any application can listen to any new `publicationList` information by subscribing to the topic `oi4/+/+/+/+/+/pub/publicationList` and `oi4/+/+/+/+/+/pub/publicationList/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific `resourceType[/<tag>]`:
`oi4/<serviceType>/<appId>/pub/publicationList/<oi4Identifier>/<resourceType>[/<tag>]`

Publish all DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/publicationList/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/publicationList`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "mam",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
          "resource": "mam",
          "subResource": "<oi4Identifier>",
          "filter": "",
          "DataSetWriterId": <UINT16>,
          "oi4Identifier": <oi4Identifier>,
          "mode": "<<enumName>_<enumValue>>",
          "interval": <UINT32>,
          "precisions": {
            "<key m of dataSet>": <REAL>,
            "<key n of dataSet>": <REAL>
          },
          "config": "<<enumName>_<enumValue>>"
      }
    },
    {
      "DataSetWriterId": <UINT16>,
      "filter": "health",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
          "resource": "health",
          "subResource": "<oi4Identifier>",
          "filter": "",
          "DataSetWriterId": <UINT16>,
          "oi4Identifier": <oi4Identifier>,
          "mode": "<<enumName>_<enumValue>>",
          "interval": <UINT32>,
          "precisions": {
            "<key m of dataSet>": <REAL>,
            "<key n of dataSet>": <REAL>
          },
          "config": "<<enumName>_<enumValue>>"
      }
    },
    {
      "DataSetWriterId": <UINT16>,
      "filter": "event",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
          "resource": "event",
          "subResource": "<event category>",
          "filter": "<event level>",
          "DataSetWriterId": <UINT16>,
          "oi4Identifier": <oi4Identifier>,
          "mode": "<<enumName>_<enumValue>>",
          "interval": <UINT32>,
          "precisions": {
            "<key m of dataSet>": <REAL>,
            "<key n of dataSet>": <REAL>
          },
```

```
            "config": "<<enumName>_<enumValue>>"
        },
      {
        "DataSetWriterId": <UINT16>,
        "filter": "data",
        "subResource": "<oi4Identifier>",
        "Timestamp": "<DateTime>",
        "Payload": {
            "resource": "data",
            "subResource": "<oi4Identifier>",
            "filter": "<tag>",
            "DataSetWriterId": <UINT16>,
            "oi4Identifier": <oi4Identifier>,
            "mode": "<<enumName>_<enumValue>>",
            "interval": <UINT32>,
            "precisions": {
              "<key m of dataSet>": <REAL>,
              "<key n of dataSet>": <REAL>
            },
            "config": "<<enumName>_<enumValue>>"
        }
      }
    ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

Set publicationList from specific asset

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subRecource>/<filter>` combination `set/publicationList/<oi4Identifier>/<resourceType>[/<tag>]` it is possible to modify publication behavior of listed publications. After a `set` message arrived, the application will publish an `event` message with status information about the `set` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `set` *must be done explicit. That means* `set/publicationList` *is only possible by using a complete topic containing* `resource`*,* `subresource` *and* `filter`*, such as* `oi4/<serviceType>/<appId>/set/publicationList/<oi4Identifier>/<resourceType>[/<tag>]`*. Setting multiple DataSetMessages at once via* `oi4/<serviceType>/<appId>/set/publicationList` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

If setting `publicationList` was successful, the new `publicationList` will be published to the `pub/publicationList/<oi4Identifier>/<resourceType>[/<tag>]` topic, according the the settings in publicationList (9.3.11).

Publisher topic:

Set explicit DataSetMessage for a specific `resourceType[/<tag>]`:
`oi4/<serviceType>/<appId>/set/publicationList/<oi4Identifier>/<resourceType>[/<tag>]`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<resourceType>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        {
          "resource": "<resource>",
          "subResource": "<subResource>",
          "filter": "<filter>",
          "DataSetWriterId": <UINT16>,
          "oi4Identifier": <oi4Identifier>,
          "mode": "<<enumName>_<enumValue>>",
          "interval": <UINT32>,
          "precisions": {
            "<key m of dataSet>": <REAL>,
            "<key n of dataSet>": <REAL>
          },
          "config": "<<enumName>_<enumValue>>"
        }
      }
    }
  ]
}
```

NOTE *More advance examples can be found in the appendix B1.1.11.*

10.1.12 subscriptionList

The resource `subscriptionList` represents the actual available/configured subscriptions in a single application and its rudimentary settings.

The `subscriptionList` contains an array of DataSetMessage objects, each representing a subscription and contains a topic path an interval for application internal usage and its configurability.

Each time a DataSetMessage or a list of DataSetMessages is added/deleted/reconfigured, the `subscriptionList` is (re)published. If the publication is a response to a reconfiguration, only the changes along with the CorrelationId from the call that triggered the reconfiguration may be published.

NOTE *All DataSetMessages of resource* `subscriptionList` *are reusing the same DataSetWriterId, because each subscriptionList entry has the same DataSet.*

The payload of a DataSetMessage (9.2.3) containing `subscriptionList` information is described in detail in section 9.3.12.

For the resource `subscriptionList`, the methods `pub`, `get`, `set` and `del` are available.

- In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `subscriptionList` object is not included. The `subResource` can be used to request only `subscriptionList` objects of a specific asset. The `filter` can be used to request `subscriptionList` objects of a specific `resourceType`. Without `subResource` and `filter`, all `publicationList` objects will be requested.

  NOTE *Depending on the resource to filter for, the* `filter` *can be* `<resourceType>` *or* `<resourceType>/<tag>`*.*
  `<resourceType>` *filters are relevant for resources without additional filter definitions (e.g. mam, health and others).* `<resourceType>/<tag>` *filters are relevant for resources, which do have filter definitions such as data, config and others.*

- In combination with the method `pub`, the payload contains DatsSetMessages of type `subscriptionList` and optional `pagination` (9.3.15).
- In combination with the method `set`, the payload contains a single DataSetMessages of type `subscriptionList`. A topic with method `set` must contain `resource`, `subResource` and `filter` to be valid.
- In combination with the method `del`, the payload contains a single DataSetMessages of type `subscriptionList`. A topic with method `del` must contain `resource`, `subResource` and `filter` to be valid.

In combination with a `resourceType` (8.1.7) it is possible to get a `subscriptionList` filtered for a defined resource. In addition to the `resourceType` a `tag` can be added, e.g. to address a defined DataSetMessage of the resource `data`.

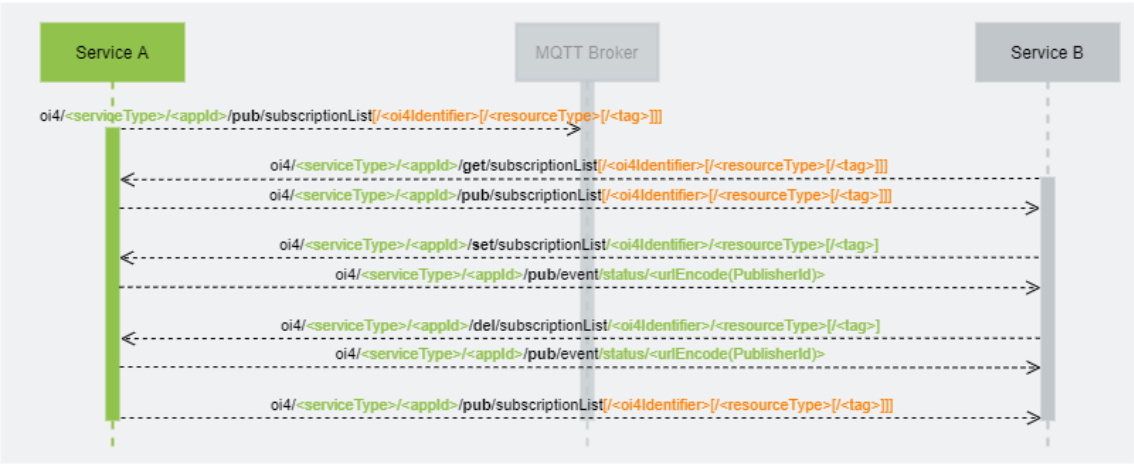

Figure 29 Sequence diagram of the four basic interactions that can be used for the resource subscriptionList

Get subscriptionList from assets

Via the `<method>/<resource>[/<subResource>[/<filter>]]` combination `get/subscriptionList[/<oi4Identifier>[/<resourceType>[/<tag>]]]`, the publication of this information to the `pub/subscriptionList[/<oi4Identifier>[/<resourceType>[/<tag>]]]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific `resourceType[/<tag>]`:
`oi4/<serviceType>/<appId>/get/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]`

Get all DataSetMessages for a specific asset:
`oi4/<serviceType>/<appId>/get/subscriptionList/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/subscriptionList`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/subscriptionList` *triggers a publication of all available* `subscriptionList` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/subscriptionList`.

Publish subscriptionList from assets

The full information is provided via the `<method>/<resource>` combination `pub/subscriptionList`. If filtered information are wanted, the `<method>/<resource>/<subResource>[/<filter>]` combination `pub/subscriptionList/<oi4Identifier>[/<resourceType>[/<tag>]]` might be used.

Any application can listen to any new `publicationList` information by subscribing to the topic `oi4/+/+/+/+/+/pub/subscriptionList` and `oi4/+/+/+/+/+/pub/subscriptionList/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific `resourceType[/<tag>]`:
`oi4/<serviceType>/<appId>/pub/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]`

Publish all DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/subscriptionList/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/subscriptionList`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<resourceType>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "topicPath": "<topic>",
        "interval": <UINT32>,
        "config": "<<enumName>_<enumValue>>"
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic, using pagination object for telling so, till all DataSetMessages are published.*

Set subscriptionList from specific asset (create new entry or set new behavior)

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subRecource>/<filter>` combination `set/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]` it is possible to modify subscription behavior of listed subscriptions. After a `set` message arrived, the application will publish an `event` message with status information about the `set` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `set` *must be done explicit. That means* `set/subscriptionList` *is only possible by using a complete topic containing* `resource`, `subresource` *and* `filter`, *such as* `oi4/<serviceType>/<appId>/set/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]`. *Setting multiple DataSetMessages at once via* `oi4/<serviceType>/<appId>/set/subscriptionList` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

If setting `subscriptionList` was successful, the new `subscriptionList` will be published to the `pub/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]` topic, according the the settings in publicationList (9.3.11).

Publisher topic:

Set explicit DataSetMessage for a specific `resourceType[/<tag>]`: `oi4/<serviceType>/<appId>/set/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<resourceType>",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "topicPath": "<topic>",
        "interval": <UINT32>,
        "config": "<<enumName>_<enumValue>>"
      }
    }
  ]
}
```

Delete subscriptionList

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subRecource>/<filter>` combination `del/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]` it is possible to delete subscriptions from the list. After a `del` message arrived, the application will publish an `event` message with status information about the `del` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `del` *must be done explicit. That means* `del/subscriptionList` *is only possible by using a complete topic containing* `resource`, `subresource` *and* `filter`, *such as* `oi4/<serviceType>/<appId>/del/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]`. *Deleting multiple DataSetMessages at once via* `oi4/<serviceType>/<appId>/del/subscriptionList` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

Publisher topic:

Delete explicit DataSetMessage for a specific `resourceType[/<tag>]`:
`oi4/<serviceType>/<appId>/del/subscriptionList/<oi4Identifier>/<resourceType>[/<tag>]`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "<resourceType>"
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "topicPath": "<topic>",
        "interval": <UINT32>,
        "config": "<<enumName>_<enumValue>>"
      }
    }
  ]
}
```

NOTE *More advance examples can be found in the appendix B1.1.12.*

10.1.13 interfaces

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

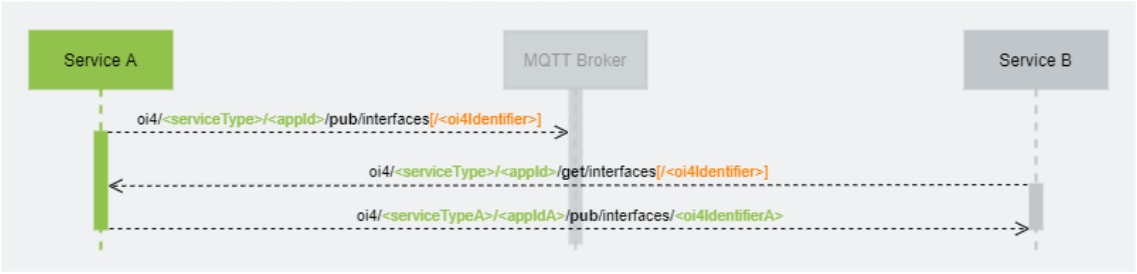

Figure 30: Sequence diagram of the two basic interactions that can be used for the resource interfaces

NOTE *More advance examples can be found in the appendix B1.1.13.*

10.1.14 referenceDesignation

The resource `referenceDesignation` represents the actual reference designation of a single asset according to IEC 81346-1:2009-07.

The payload of a DataSetMessage (9.3.14) containing `referenceDesignation` information is described in detail in section 9.3.14.

For the resource `referenceDesignation`, the methods `pub`, `get`, `set` and `del` are available.

- In combination with the method `get`, the payload contains only optional DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `referenceDesignation` object is not included. The requested information is identified via the filter contained in the topic. If neither the oi4Identifier nor

DataSetWriterId is specified as filter, all config information are requested.
In combination with the method `get`, the payload requires an empty NetworkMessage. The NetworkMessage can alternatively contain DataSetMessages for `pagination` (9.3.15) and `locale` (9.3.16). A `referenceDesignation` object is not included. The `subResource` can be used to request only the `referenceDesignation` objects of a specific asset. Without `subResource`, all `referenceDesignation` objects will be requested.

- In combination with the method `pub`, the payload contains DatsSetMessages of type `referenceDesignation` and optional `pagination` (9.3.15).
- In combination with the method `set`, the payload contains a single DataSetMessages of type `referenceDesignation` (9.3.15). A topic with method `set` must contain `resource` and `subResource` to be valid.
- In combination with the method `del`, a `referenceDesignation` object is not included. The requested information is identified via the oi4Identifier contained in the topic. A topic with method `del` must contain `resource` and `subResource` to be valid.

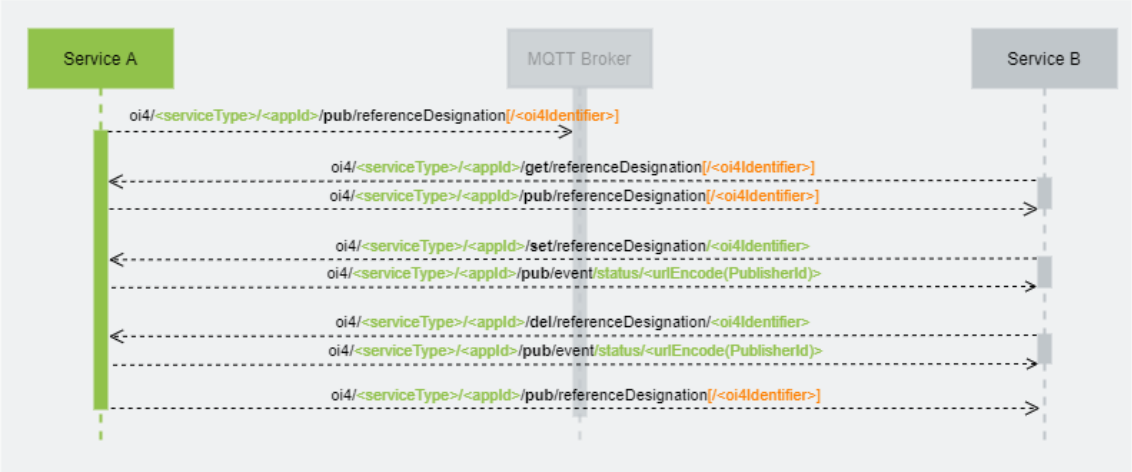

Figure 30 Sequence diagram of the interactions that can be used for the resource referenceDesignation

Get referenceDesignation from assets

Via the `<method>/<resource>[/<subResource>]` combination `get/referenceDesignation[/<oi4Identifier>]`, the publication of this information to the `pub/referenceDesignation[/<oi4Identifier>]` topic can be triggered.

Publisher topic:

Get explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/get/referenceDesignation/<oi4Identifier>`

Get DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/get/referenceDesignation`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *A topic such as* `oi4/<serviceType>/<appId>/get/referenceDesignation` *triggers a publication of all available* `referenceDesignation` *information, each in its own DataSetMessage added to a NetworkMessage and published to the topic* `oi4/<serviceType>/<appId>/pub/referenceDesignation`*.*

Publish referenceDesignation

The full information is provided via the `<method>/<resource>` combination `pub/referenceDesignation`. If filtered information are wanted, the `<method>/<resource>/<subResource>` combination `pub/referenceDesignation/<oi4Identifier>` might be used.

Any application can listen to any new `referenceDesignation` information by subscribing to the topic `oi4/+/+/+/+/+/pub/referenceDesignation` and `oi4/+/+/+/+/+/pub/sreferenceDesignation/#`.

Publisher topic:

Publish explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/pub/referenceDesignation/<oi4Identifier>`

Publish DataSetMessages for all assets:
`oi4/<serviceType>/<appId>/pub/referenceDesignation`

NOTE *Please keep in mind, that a NetworkMessage might be spitted via pagination mechanism into several messages.*

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "location": {
          "value": "<designation value>",
          "local": "<local designation value>",
          "parent": {
            "value": "<designation value>",
            "local": "<local designation value>",
            "oi4Identifier": "<oi4_identifier>"
          }
        },
        "function": {
          "value": "<designation value>",
          "local": "<local designation value",
          "parent": {
            "value": "<designation value>",
            "local": "<local designation value>",
            "oi4Identifier": "<oi4_identifier>"
          }
        },
        "product": {
          "value": "<designation value>",
          "local": "<local designation value>",
          "parent": {
            "value": "<designation value>",
            "local": "<local designation value>",
            "oi4Identifier": "<oi4_identifier>"
          }
        }
      }
    }
  ]
}
```

NOTE *Once, too many DataSetMessages are available to add to a single NetworkMessage, pagination mechanism must be used (9.3.15). The publisher publishes as many NetworkMessages to the same topic as specified in the pagination object, till all DataSetMessages are published.*

Set referenceDesignation from specific asset

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subRecource>` combination `set/referenceDesignation/<oi4Identifier>` it is possible to modify the reference designation. After a `set` message arrived, the application will publish an `event` message with

status information about the `set` command to the topic `pub/event/status/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `set` *must be done explicit. That means* `set/referenceDesignation` *is only possible by using a complete topic containing* `resource` *and* `subresource` *, such as* `oi4/<serviceType>/<appId>/set/referenceDesignation/<oi4Identifier>`*. Setting multiple reference designations at once via* `oi4/<serviceType>/<appId>/set/referenceDesignation` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

If setting `referenceDesignation` was successful, the new `referenceDesignation` will be published to the `pub/referenceDesignation/<oi4Identifier>/<resourceType>[/<tag>]` topic, according the the settings in publicationList (9.3.11).

Publisher topic:

Set explicit DataSetMessage for a specific asset `oi4/<serviceType>/<appId>/set/referenceDesignation/<oi4Identifier>`

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "filter": "",
      "subResource": "<oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "location": {
          "value": "<designation value>",
          "local": "<local designation value>",
          "parent": {
            "value": "<designation value>",
            "local": "<local designation value>",
            "oi4": "<oi4_identifier>"
          }
        },
        "function": {
          "value": "<designation value>",
          "local": "<local designation value",
          "parent": {
            "value": "<designation value>",
            "local": "<local designation value>",
            "oi4": "<oi4_identifier>"
          }
        },
        "product": {
          "value": "<designation value>",
          "local": "<local designation value>",
          "parent": {
            "value": "<designation value>",
            "local": "<local designation value>",
            "oi4": "<oi4_identifier>"
          }
        }
      }
    }
  ]
}
```

Delete referenceDesignation from specific asset

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<method>/<resource>/<subRecource>` combination `del/referenceDesignation/<oi4Identifier>` it is possible to delete a reference designation. After a `del` message arrived, the application will publish an `event` message with status information about the `del` command to the topic `pub/event/<oi4>/<urlEncode(PublisherId)>` (9.3.9.1) to provide feedback to the caller.

NOTE *Each publication, using the method* `del` *must be done explicit. That means*

`del/referenceDesignation` *is only possible by using a complete topic containing* `resource` *and* `subresource`*, such as* `oi4/<serviceType>/<appId>/del/referenceDesignation/<oi4Identifier>`*. Deleting multiple reference designations at once via* `oi4/<serviceType>/<appId>/del/referenceDesignation` *is not possible due to security reasons because it can not be validated via an access control list by the Message Bus.*

Publisher topic:

Delete explicit DataSetMessage for a specific asset:
`oi4/<serviceType>/<appId>/del/referenceDesignation/<oi4Identifier>`

Payload:

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
```

NOTE *More advance examples can be found in the appendix B1.1.14.*

10.2 Common services

All services, fully defined in Open Industry 4.0 Alliance context are described in the following sub-chapters. They are communicating in a call/reply pattern.

Unlike the specific services (10.3), these services have well-defined input and output arguments and can be used without modification for any Alliance-compliant application that supports these services.

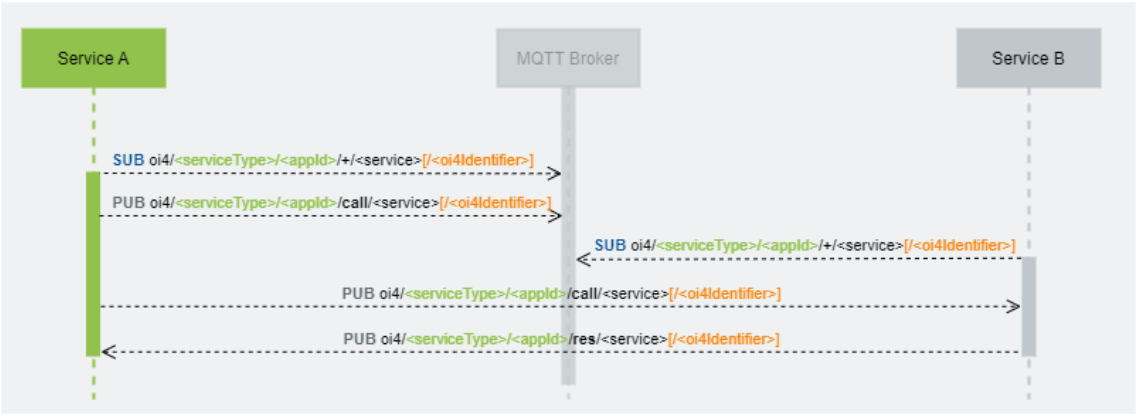

Figure 31 Sequence diagram of the interactions on how to use method calls over the Message Bus

10.2.1 fileUpload

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The service `fileUpload` uploads a file to an application or to a device. The `inputArguments` and the `outputArguments` are described in section 9.4.1.

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

This common function cannot check whether the file for upload may be transported or not. The legitimacy of the file must be checked by the application after receiving the file.

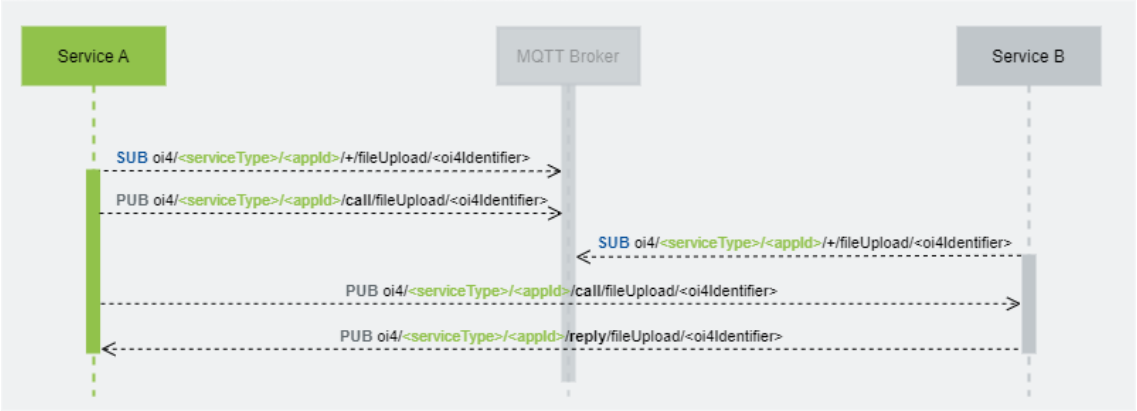

Figure 32 Sequence diagram of the interactions on how to use the method call fileUpload over the Message Bus

The information is provided via the `<method>/<services>/<subResource>` combination `call/fileUpload/<oi4Identifier>` to request a file and `reply/fileUpload/<oi4Identifier>` to reply the file and/or status.

Publisher topic for call:

```
oi4/<serviceType>/<appId>/call/fileUpload/<oi4Identifier>
```

Call

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "3b4a62ba-026f-4ee8-bc99-3a5f85fc9f3b",
  "correlationId ": "<empty/not present> or <initial MessageId>",
  "Message": {
    "methodsToCall": [
      {
        "methodId": "fileUpload",
        "inputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

Publisher topic for reply:

```
oi4/<serviceType>/<appId>/reply/fileUpload/<oi4Identifier>
```

Reply

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "3b4a62ba-026f-4ee8-bc99-3a5f85fc9f3b",
  "correlationId ": "<Caller MessageId>",
  "Message": {
    "results": [
      {
        "statusCode": "<StatusCode>",
        "inputArgmentResults": ["<StatusCode>"],
        "outputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

10.2.2 fileDownload

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The service `fileDownload` downloads a file from an application or from a device. The `inputArguments` and the `outputArguments` are described in section 9.4.2.

RECOMMENDATION *To prevent access by unauthorized communication participants, the operator must prevent this via ACL.*

This common function cannot check whether the file for download may be transported or not. The legitimacy of the file must be checked by the application which requested the file after receiving it.

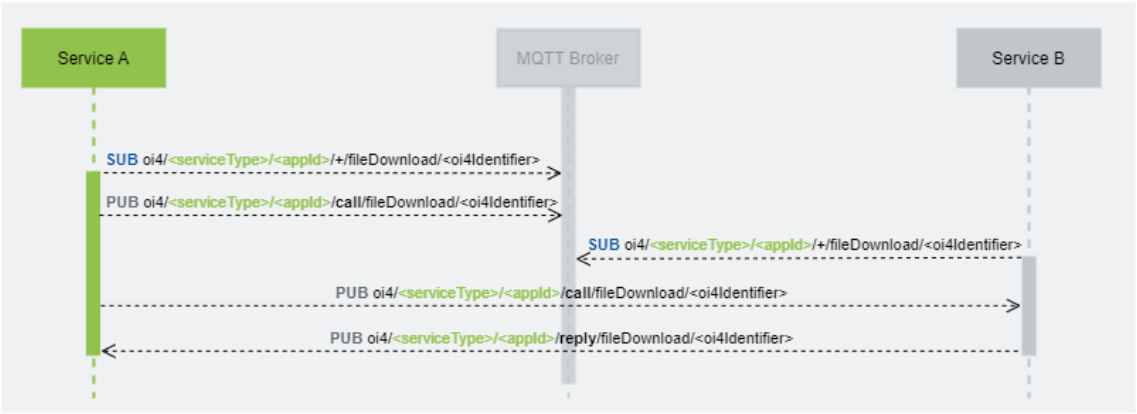

Figure 33 Sequence diagram of the interactions on how to use the method call fileDownload over the Message Bus

The information is provided via the `<method>/<services>/<subResource>` combination `call/fileDownload/<oi4Identifier>` to request a file and `reply/fileDownload/<oi4Identifier>` to reply the file and/or status.

Publisher topic for call:

`oi4/<serviceType>/<appId>/call/fileDownload/<oi4Identifier>`

Call

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "760abda2-ba40-4e6e-863a-eea8c002b4e4",
  "correlationId ": "<empty/not present> or <initial MessageId>",
  "Message": {
    "methodsToCall": [
      {
        "methodId": "fileDownload",
        "inputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

Publisher topic for reply:

`oi4/<serviceType>/<appId>/reply/fileDownload/<oi4Identifier>`

Reply

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "760abda2-ba40-4e6e-863a-eea8c002b4e4",
  "correlationId ": "<Caller MessageId>",
  "Message": {
    "results": [
      {
        "statusCode": "<StatusCode>",
        "inputArgmentResults": ["<StatusCode>"],
        "outputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

10.2.3 firmwareUpdate

ATTENTION *Information in this chapter requires alignment with other working groups and has*

not been maturely specified.

The service `firmwareUpdate` uploads a firmware to an application or to a device. The `inputArguments` and the `outputArguments` are described in section 9.4.3.

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

This common function cannot check whether the firmware is valide or not. The legitimacy of the firmware must be checked by the application after receiving it. Additionally, technology depended state machines for update procedures might be necessary.

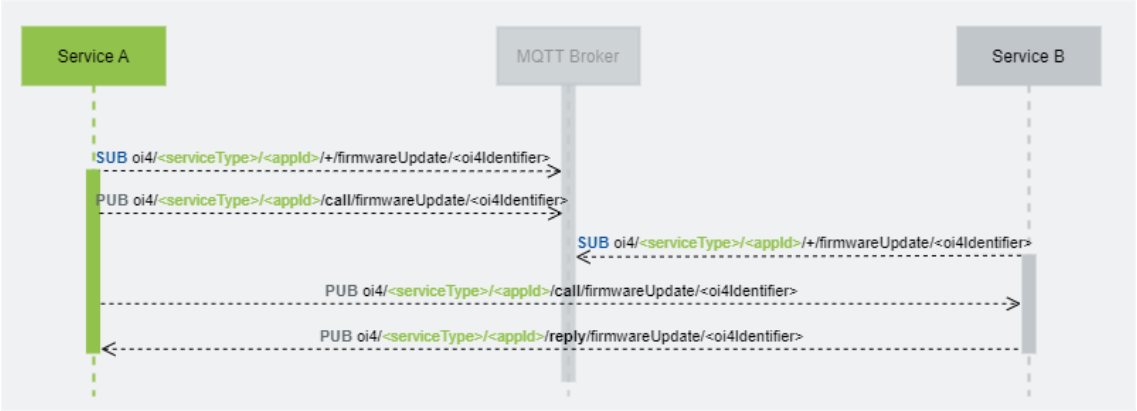

Figure 34 Sequence diagram of the interactions on how to use the method call firmwareUpdate over the Message Bus

The information is provided via the `<method>/<services>/<subResource>` combination `call/firmwareUpdate/<oi4Identifier>` to request an update and `reply/firmwareUpdate/<oi4Identifier>` to reply the status of the update.

Publisher topic for call:

`oi4/<serviceType>/<appId>/call/firmwareUpdate/<oi4Identifier>`

Call

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "414e26f6-341b-43b7-90fc-bb9e0b1b0866",
  "correlationId ": "<empty/not present> or <initial MessageId>",
  "Message": {
    "methodsToCall": [
      {
        "methodId": "firmwareUpdate",
        "inputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

Publisher topic for reply:

`oi4/<serviceType>/<appId>/reply/firmwareUpdate/<oi4Identifier>`

Reply

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "414e26f6-341b-43b7-90fc-bb9e0b1b0866",
  "correlationId ": "<Caller MessageId>",
  "Message": {
    "results": [
      {
        "statusCode": "<StatusCode>",
        "inputArgmentResults": ["<StatusCode>"],
        "outputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

10.2.4 blink

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The service `blink` requests a blink signal from device talking to. The `inputArguments` and the `outputArguments` are described in section 9.4.4.

Most field buses and real-time ethernet systems are having such a service to identify devices in the field.

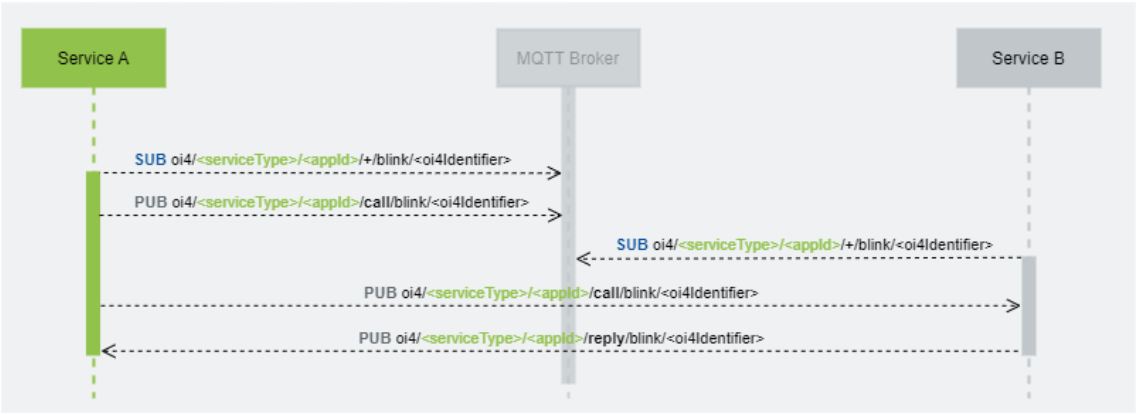

Figure 35 Sequence diagram of the interactions on how to use the method call blink over the Message Bus

The information is provided via the `<method>/<services>/<subResource>` combination `call/blink/<oi4Identifier>` to request a device to blink and `reply/blink/<oi4Identifier>` to reply if method was successful.

Publisher topic for call:

```
oi4/<serviceType>/<appId>/call/blink/<oi4Identifier>
```

Call

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "3b423a40-a676-4ba0-8017-f0b2cd65bc26",
  "correlationId ": "<empty/not present> or <initial MessageId>",
  "Message": {
    "methodsToCall": [
      {
        "methodId": "blink",
        "inputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

Publisher topic for reply:

```
oi4/<serviceType>/<appId>/reply/blink/<oi4Identifier>
```

Reply

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "3b423a40-a676-4ba0-8017-f0b2cd65bc26",
  "correlationId ": "<Caller MessageId>",
  "Message": {
    "results": [
      {
        "statusCode": "<StatusCode>",
        "inputArgmentResults": ["<StatusCode>"],
        "outputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

10.2.5 newDataSetWriterId

In some cases it is required to set a new DataSetMessage in an Application from outside the application. E.g. a reference designation (9.3.14), known by a MES system shall be written into an OTConnector. Therefore the ITConnector, dealing with the MES system, has to set/create the referenceDesignation inside the OTConnector. To do so, a valid and unique DataSetWriterId for the OTConnector is necessary.

The service `newDataSetWriterId` requests a new, so far unused, DataSetWriterId from the Publisher, which consumes this method call. The `inputArguments` and the `outputArguments` are described in section 9.4.5.

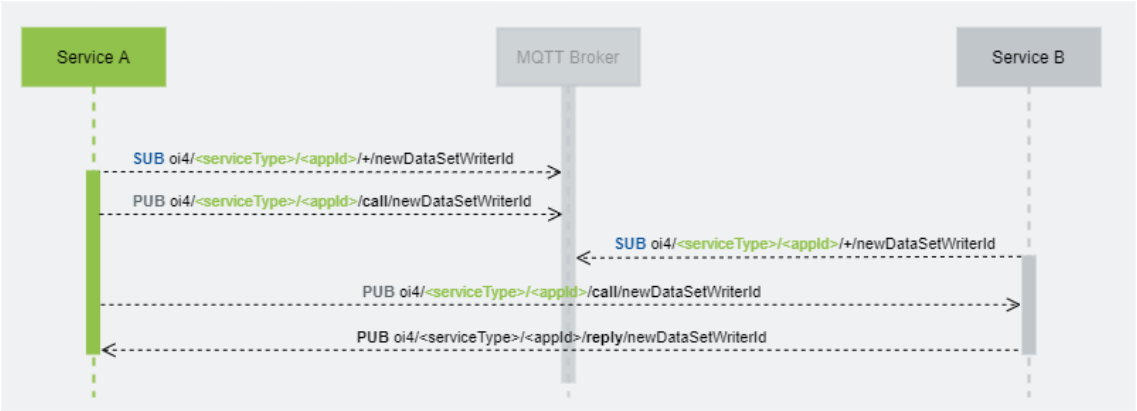

Figure 36 Sequence diagram of the interactions on how to use the method call newDataSetWriterId over the Message Bus

The information is provided via the `<method>/<services>` combination `call/newDataSetWriterId` to request and `reply/newDataSetWriterId` to reply the new DataSetWriterId.

Publisher topic for call:

`oi4/<serviceType>/<appId>/call/newDataSetWriterId`

Call

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "2aca55bd-0d6f-41b1-a1c2-2d61afcc21f0",
  "correlationId ": "<empty/not present> or <initial MessageId>",
  "Message": {
    "methodsToCall": [
      {
        "methodId": "newDataSetWriterId",
        "inputArguments": [
          {
            "resource": "<resourceType>"
          }
        ]
      }
    ]
  }
}
```

Publisher topic for response:

```
oi4/<serviceType>/<appId>/reply/newDataSetWriterId
```

Reply

```
{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<serviceType>/<appId>",
  "DataSetClassId ": "2aca55bd-0d6f-41b1-a1c2-2d61afcc21f0",
  "correlationId ": "<Caller MessageId>",
  "Message": {
    "results": [
      {
        "statusCode": <UINT32>,
        "inputArgmentResults": [<UINT32>],
        "outputArguments": [
          {
            "DataSetWriterId": <UINT16>,
            "ttl": <UINT32>                    //in seconds
          }
        ]
      }
    ]
  }
}
```

10.3 Specific services

All services, rudimentarily defined in Open Industry 4.0 Alliance context are described in the following sub-chapters. They are communicating in a call/reply pattern.

In opposite to common services (10.2), these services have a defined service name, but application specific input and output arguments.

Figure 32 shows a sequence diagram of the interactions on how to use method calls over the Message Bus.

10.3.1 read

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The service `read` returns the data, which were requested.

RECOMMENDATION *To prevent access by unauthorized communication participants, the operator must prevent this via ACL.*

Because every underlying protocol and every application is different, there is no way to define the input and output arguments in a common way - only the service name in the used topic should be defined.

10.3.2 write

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The service `write` writes the given data to a defined asset.

RECOMMENDATION *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Because every underlying protocol and every application is different, there is no way to define the input and output arguments in a common way - only the service name in the used topic should be defined.

10.3.3 subscribe

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The `subscribe` service subscribes to a defined data endpoint and returns its content if it is changed.

Because every underlying protocol and every application is different, there is no way to define the input and output arguments in a common way - only the service name in the used topic should be defined.

10.3.4 unsubscribe

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The `unsubscribe` service unsubscribes a previously subscribed data endpoint.

Because every underlying protocol and every application is different, there is no way to define the input and output arguments in a common way - only the service name in the used topic should be defined.

10.3.5 genericMethod

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

The placeholder `genericMethod` is not a service itself. The placeholder is used in the call/reply topic, while the method name and input and output parameters are defined in the payload.

RECOMMENDATION *To prevent access by unauthorized communication participants, the operator must prevent this via ACL.*

The placeholder is used to implement all non standardized services.

Because every underlying protocol and every application is different, there is no way to define the input and output arguments in a common way - only the service name in the used topic should be defined.

11 The Open Edge Computing Platform (MVP)

Open Edge Computing platform scenarios compliant to the Industry 4.0 Alliance are shown in figure 38. They have typical characteristics:

- Consisting of at least one Edge Devices
- Containing an operating system including basic OI4 services
- Containing a docker daemon
- Containing an MQTT broker utilized as Message Bus between applications and devices
- Applying containerized applications to fulfill customers requirements

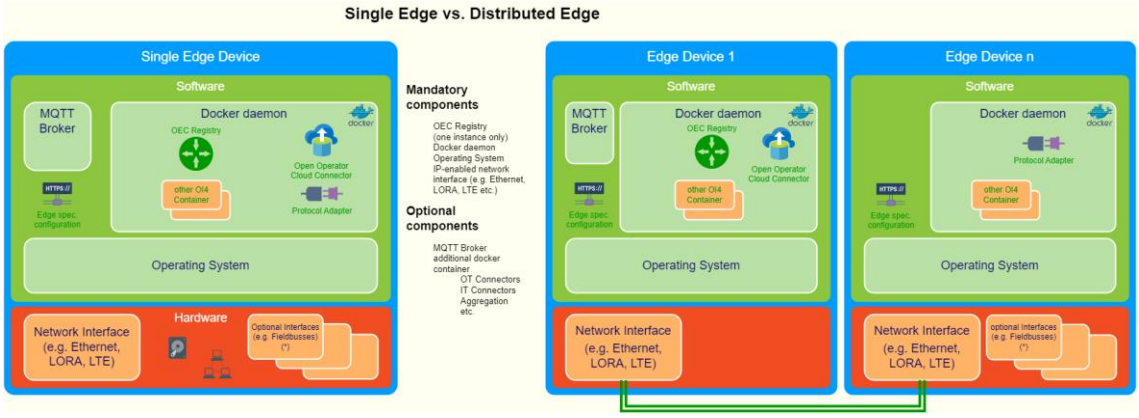

Figure 37 Block diagram of possible Open Edge Computing architectures

To label a device, system or application as “Open Industry 4.0 Alliance Compliant” (see figure 39), several compliance requirements must be fulfilled.

Figure 38 Badge for Alliance compliant products

The following subchapters provide an overview about the compliance requirements. For the full list of requirements, the Alliance offers a document on request called *Requirements for Interoperability*.

11.1 Minimum requirements on an Alliance compliant OEC platform

The heart of an Open Edge Computing platform is the Edge Device with its installed software components. To achieve Open Industry 4.0 Alliance Compliance, several requirements must be met by the hardware and software.

From the Alliance perspective, there is a set of requirements that defines the minimum class of edge devices. A general overview of these requirements is given here.

The hardware requirements are:

- The actual supported processor architectures are ARMv7, ARMv8 and amd64
- A minimum of 1 GB RAM
- A minimum of 4 GB non volatile memory (eMMC, SD Card, SSD, HDD, …)
- At least one available network interface (likely Ethernet interface, but LTE, etc. might be possible too)
- The system should be hardened for industrial use

The software requirements are:

- The operating system shall be a Linux-based distribution based on the current patch level.
- The user interface of an Open Edge Computing platform provides a user and role management for access control
- The authentication methods must fulfill industry grade standards
- Access to available services of the host system can be granularly configured via the user and role management.
- A Docker daemon (6) must be integrated and available for Open Edge Computing users
- A MQTT broker (7) must be available on the Edge Device or within the Open Edge Computing platform.
- An OEC Registry (A1) must be available on the Edge Device or within the Open Edge Computing platform.

For a more detailed overview of the requirements to be met, please refer to the external document *Requirements for Interoperability*.

11.2 Minimum requirements on an Alliance compliant application

To achieve seamless application- and vendor-interoperability, the Alliance defines compliance requirements for docker-based applications. For a detailed overview, please refer to the external document *Requirements for Interoperability*.

A general overview of the requirements is given here:

- Provided container images shall be available for at least one of the supported platforms. Preferred are multi architecture images.
- The container environment specific requirements need to be fulfilled (6).
- The Message Bus communication requirements must be fulfilled (see 7).
- The minimum set of methods to be used over the Message Bus must be supported (8.2)

- Each application should describe its minimum hardware requirements
- Each application shall list all internally used licenses
- Services offered in applications must follow legal guidelines and not contain harmful or misusing components (like, crypto miner, spy software, trojan horses, viruses, etc.).
- Each application should be developed according to common rules for good software engineering and quality assurance

Besides the connection to the Message Bus, applications might provide interfaces to users, systems or other applications. Typical additional interfaces are UIs (User Interface), web services and shopfloor protocols. The access to the interfaces might need authentication and authorization for security reasons, like basic authentication, OAuth or client certificate-based authentication. Since this is not part of the Alliance' guideline definitions, these authentication and authorization mechanisms reside with the application itself.

11.3 Legal requirements, software licenses

Services (and/or container images) often contain third-party software. It is in the full responsibility of the provider that compliance with the associated software licenses must be fulfilled. Every license used within a service must be provided with the use of the license resource explained in chapter 9.3.4.

Utilized procedures and techniques within the services protected by any patent or intellectual property that does not belong to the service provider require a valid patent usage or licence agreement.

The offered services may only operate within the boundaries of the applicable law and must not support unethical tasks, like crypto-mining or spy software.

Appendix A

A1 The OEC Registry

Beside the Message Bus, the OEC Registry is one of the basic components, which an Open Industry 4.0 Alliance compliant Open Edge Computing platform consists of.

It collects the data of defined Open Industry 4.0 Alliance resources from the Message Bus and provides it via Message Bus and web front end to others.

The OEC Registry's main tasks are:

- Collect information about reachable assets (applications and devices)
- Provide Master Asset Models of all available assets (applications and devices) to others on the Message Bus
- Provide a web front end, which shows relevant information to its users
    - List all running Open Industry 4.0 Alliance applications
        - incl. nameplate of the application (10.1.1 mam (Master Asset Model))
        - incl. last health information (10.1.2 health) and availability
        - incl. supported resources (10.1.10 profile)
    - List all detected devices
        - incl. nameplate of the device (10.1.1 mam (Master Asset Model))
        - incl. last health information (10.1.2 health) and availability
        - incl. supported resources (10.1.10 profile)
        - incl. related publisher application
    - Show the audit trail, which was published via event resource to the Message Bus (10.1.9 event)

Because of its central position it might offer additional services such as:

- Store audit trail to log file to make it available offline (support relevant).
- Check basic conformity, based on schema validation of received messages, and show the result.
- Active extended conformity checks, based on schema validation of additional subscriptions to supported resources.

For a seamless operation the OEC Registry should be the application, which gets started first on the Open Edge Computing platform.

NOTE *A reference implementation of the OEC Registry is available free of charge and can be used by any member of the Open Industry 4.0 Alliance. It comes without any warranty and the source code is available under different licenses.*
The default ports for the web front end and the REST API to its back end are 5798 and 5799.
The OEC Registry communicates via TLS. The certificate should be placed in a given path, otherwise a self signed certificate is used.

A2 Predefined DataSetClassIds for resources of the Alliance

The `DataSetClassId` is a PubSub parameter (OPC UA Part 14-6.2.2.2) and used to identify globally define DataSet classes - in opposite to locally defined DataSets, e.g. through the publisher.

The `DataSetClassId` is an optional key for the objects DataSetMetaData (9.2.2), NetworkMessage (9.2.1) and ServiceNetworkMessage (9.2.16). It is of type `GUID` (OPC UA Part 6-5.1.3), which is represented in JSON as a string with separator (OPC UA Part 6-5.4.2.7).

All standardized Open Industry 4.0 Alliance resources (8.1.5) have a defined DataSet and therefore a globally unique `DataSetClassId`, which are listed here:

resource (DataSet described in 9.3 ff)	**DataSetClassId**
mam (9.3.1)	360ca8f3-5e66-42a2-8f10-9cdf45f4bf58
health (9.3.2)	d8e7b6df-42ba-448a-975a-199f59e8ffeb
config (9.3.3)	9d5983db-440d-4474-9fd7-1cd7a6c8b6c2
license (9.3.4)	2ae0505e-2830-4980-b65e-0bbdf08e2d45
licenseText (9.3.5)	a6e6c727-4057-419f-b2ea-3fe9173e71cf
rtLicense (9.3.6)	ebd12d4b-da1c-4671-ab86-db102fecc603
event (9.3.9)	543ae05e-b6d9-4161-a0a3-350a0fac5976
profile (9.3.10)	48017c6a-05c8-48d7-9d85-4b08bbb707f3
publicationList (9.3.11)	217434d6-6e1e-4230-b907-f52bc9ffe152
subscriptionList (9.3.12)	e5d68c47-c276-4929-8ab9-4c1090cac785
interfaces (9.3.13)	96d22d73-bce6-42d3-9949-45e0d04e4d54
referenceDesignation (9.3.14)	27a75019-164a-496d-a38b-90e8a55c2cfa
methods (call/reply described in 9.4 ff)	**DataSetClassId**
fileUpload (9.4.1)	3b4a62ba-026f-4ee8-bc99-3a5f85fc9f3b
fileDownload (9.4.2)	760abda2-ba40-4e6e-863a-eea8c002b4e4
firmwareUpdate (9.4.3)	414e26f6-341b-43b7-90fc-bb9e0b1b0866
blink (9.4.4)	3b423a40-a676-4ba0-8017-f0b2cd65bc26
newDataSetWriterId (9.4.5)	2aca55bd-0d6f-41b1-a1c2-2d61afcc21f0
Reserved	32c0a57a-24fd-4378-8eb3-4a42a481bf56 0721c712-f7b5-4ad0-b2a0-58f0a5744963 d08ade63-777b-464a-8f58-73649dc3ec1c 965ab782-4669-480e-bab4-c6a3b14ee4cf d2ee7674-cad5-4b59-a37c-3631bb0cd409 a9031581-4c03-4c6a-9c12-dd7fe0123ef5 04cac81f-9056-4af3-a8a0-686fc499b5f9

Table 15 Predefined DataSetClassIds for resources of the Alliance

A3 Glossary

AAS	Asset Administration Shell; The Asset Administration Shell is a concept developed by Plattform Industrie 4.0. It aims at standardizing a reference to assets in manufacturing plants. The Master Asset Model in the context of the Alliance is closely related to the AAS identification submodel.
Asset	An asset is an identifiable and relevant entity within the system. It might be a whole machine, a single physical component or a software/application. In the context of the Alliance, each asset has its own oi4Identifier und a MAM.
Birth message	A specific MQTT message type, used in the lifecycle of an MQTT client (see section 7.3.3 for details).
CCC	Common Cloud Central: The Common Cloud Central layer is one of the four high level architecture layers of the Alliance.
Close Message	A specific MQTT message type, used in the lifecycle of an MQTT client (see section 7.3.4 for details).
DIN SPEC 27070	A German standard for requirements and reference architecture of a security gateway for the exchange of industry data and services
DIN SPEC 91406	A German standard for automatic identification of physical objects and information on physical objects in IT systems, particularly IoT systems
Edge Device	An Edge Device (or Edge Gateway) that fulfills the functional criteria described in this guideline. It is located in the OEC Layer.
Field Device	An automation component with connectivity towards the Edge Layer.
IEC 62443	An international series of standards on "Industrial communication networks - IT security for networks and systems". The standard is divided into different sections and describes both technical and processor-related aspects of industrial cyber-security.
IEC 81346	An international series of standards on "Industrial systems, installations and equipment and industrial products – structuring principles and reference designations" defines the rules for reference designation systems (RDS).
mam	Master Asset Model; The Master Asset Model is a basic concept of the Alliance that provides a critical subset of information about an asset that interacts in an Alliance' context. It is closely modeled after the device information defined in OPC UA Part 100 and is described in detail in chapter 4.
NE107	NAMUR standard for Self-Monitoring and Diagnosis of Field Devices
OEC	Open Edge Computing; The Open Edge Computing layer is one of the four high level architecture layers of the Alliance. Open Edge Computing mainly deals with the containerized software environment that Alliance' compliant edge devices run.
OEC Registry	Mandatory functionality within an Open Industry 4.0 Alliance Open Edge Computing platform that collects information about connected devices, installed applications and dedicated services provided by them. The OEC Registry offers information which it gains by listening to the Message Bus.
oi4Identifier	In the Open Industry 4.0 Alliance context, unique identification of assets and communication partners is given a high priority. To this end, an oi4Identifier has been defined (see chapter 3 for details).

OOC	Open Operator Cloud; The Open Operator Cloud layer is one of the four high level architecture layers of the Alliance. The Open Operator Cloud defines the services and interfaces run on a manufacturer's IT platform for interaction with the other layers in an Alliance' compliant system.
OPC UA	OPC UA, OPC UA pubsub, OPC UA JSON, …
Process Values	In context of the Alliance, a DataSetMessage called "Process Values" (oi4_pv, see 9.3.7) is available. In contrast to Primary Value, known from Process Industry, the Process Values might have a more complex structure, containing several data combined in a DataSetMessage.
syslog	syslog is a standard for transmitting log messages (RFC 5424 - Syslog Protocol)
Will message	A specific MQTT message type, used in the lifecycle of an MQTT client (see section 7.3.5 for details).

Table 16 Glossary

A4 Authors

Author	Company
Matthias Betz	ifm electronic gmbh
Ervin Binkert	Endress+Hauser Flowtec AG
Werner Blumenstock	Siemens AG
Sebastian Czech	Harro Höfliger
Marco Colucci	Endress+Hauser Flowtec AG
Stefan Eggert	M&M Software GmbH
Martin Flöer	Weidmueller
Michael Gernoth	ifm electronic gmbh
Dr. Andreas Graf Gatterburg	Hilscher Gesellschaft für Systemautomation mbH
Thomas Greil	Voith
Hitoshi Hattori	Yokogawa
Konrad Heidrich	Hilscher Gesellschaft für Systemautomation mbH
Michael Heller	M&M Software GmbH
Hans-Jürgen Hilscher	Hilscher Gesellschaft für Systemautomation mbH
Stephan Huber	Endress+Hauser Process Solutions (DE) GmbH
Daniel Kerkhoff	Beckhoff
Daria Khorkunova	Fujitsu
Francois Klein	Endress+Hauser Flowtec AG
Christopher Kleinert	MULTIVAC
Vitas Kling	Dunkermotoren
Peter Kob	Siemens AG
Alexander Lehmann	Hensoldt Cyber GmbH
Matthias Lempertseder	CAPTRON
Artur Loorpuu	UReason
Lucas Mühlhaupt	Dunkermotoren
Fillipp Murroni	SAP SE
Hendrik Nieweg	Device-Insight
Elena Petrevska	Pepperl+Fuchs
Smitha Rayala	SAP SE
Markus Rentschler	Balluff

Author	Company
Masanori Sakagami	Yokogawa
Manuel Sauer	SAP SE
Christian Scherer	Awinia
Bastian Schmick	ifm electronic gmbh
Matthias Schmidt	ifm electronic gmbh
Tobias Schwartz	ifm electronic gmbh
Peter Sorowka	Cybus
Dr. Stephan Theis	nekst one
Alexander Thiel	MULTIVAC
David Weiß	ifm electronic gmbh
Dr. Thiago Weber Martins	SAP SE
Thomas Weinschenk	Endress+Hauser Process Solutions (DE) GmbH
Mathis Zeiher	Endress+Hauser Flowtec AG

Table 17 Authors (in alphabetical order)

A5 History

Revision	Date	Author	Changes/comment
0	2019-07-11	Konrad Heidrich	Initial document
0.12	2020-04-15 to 2020-12-16	Konrad Heidrich	• Due to moving/adding chapters, a renumbering was necessary • Renames Registry to OEC Registry in whole document • corrected List of tables and List of figures • Type of DataSetWriterId changed to UINT16, as defined in latest OPCF spec • Changed oi4Identifier in details, to become DIN SPEC 91406 complient (3, 3.1 and 3.2) • Corrected typo in figure 4, 5 and 6 • Chapter 4 described in more detail to be more precise • Added information to 5. • Added Registry to 5.2.1 its sequence diagram • Moved sub chapter Docker and extended/reworked it (6) • Added prefix to docker environment variables (6.2) • added max payload size to environment variables (see 6.2) • Finalized chapter *Docker Image Integrity* (6.4) • Added information to 7.1, 7.2.2 • Added Info about “other than publish immediately” to 7.4 • Added serviceType “ITConnector” (8.1.2) and improved explanations • subResource elements for resource event are renamed according to RFC5424 (8.1.6.1) • Added DataSetWriterId and additional information to 8.1.7 • Appended additional key called POI to DataSetMessage (9.1.1, 9.2.3, 10.1ff) • Appended additional key called POI to DataSetMetaData (9.1.2, 9.2.2, 10.1.8) • Append introduction to chapter 9.2 • Format of PublisherId concretized, DataSetClassId not mandatory anymore (9.2.1) • DataSetClassId set to optional in 9.2.16 • Added chapter for Alliance' defined objects (9.3) and moved information from 10.1 to it • The key healthState got changed to healthScore (9.3.2, 10.1.2) • Added implementation of config to 9.3.3 and 10.1.3

Revision	Date	Author	Changes/comment
			• The key Author got changed to Authors (9.3.4, 10.1.4) • Simplified DataSet of license object (9.3.4, 10.1.4) • Extended chapter 9.3.9 (*event*) for the categories syslog, OPC UA Status Code and NAMUR NE107 • Added pagination to 9.3.13 • Added locale to 9.3.14 • Renamed chapter 10 to *Message Bus communication* • Added sequence diagram to 10.1 and all sub-chapters for better understanding • Added information on which methods are available for which resource (10.1.x) • Added basic information to the payload structure of each resource (10.1.x) • Corrections of keys locale and text in 10.1.1 • Corrected DataSetWriterId to DataSetClassId in explanation (10.1.8). • Adopted subResource changes, according to RFC5424, to event description (10.1.9) • Added "resource" and concretized the introductory text for publicationList (10.1.11) • Changed precision definition, naming (plural) and status definition to active (10.1.11) • CREATE_2 and DELETE_4 eliminated in config enumeration of subscriptionList (10.1.12) • Reworked chapter 11 (Open Edge Computing platform) and sub chapters • Reordered complete Appendix A and Appendix B • Added Registry information (A1) • Extended glossary (A3) • Added additional examples, related to 10.1, to appendix (B1) • Moved document history to appendix (A5)
1.0.0	2021-01-11 to 2021-12-16	Konrad Heidrich	• Document title changed • Spelling and grammar improved throughout the document • Renewed several links to external content • Added *Conventions* chapter (1) to the Guideline to explain, how to read this document • Actualized whole chapter *Overall process description* (5) • Renamed and reworked whole chapter 6 to *Container environment*

Revision	Date	Author	Changes/comment
			○ Reworked container storage options (6.1 ff)* ○ Deleted mandatory environment variables* ○ Added chapter *Container networks* (6.4) • Changed broker configuration mechanism (7.2.1)* • Security measures for Message Bus specified (7.2.2)* • Clarified usage of Birth, Close and Will message with an example in 7.3.3, 7.3.4 and 7.3.5 • Reworked whole chapter 8 to simplify topic schema* • Renamed methods req/res to call/reply (8.1.4, 8.1.5, 9.4, 10, 10.2, 10.3) • Extended filter options in topic of resource publicationList (8.1.7, 10.1.11)* • Extended filter options in topic of resource subscriptionList (8.1.7, 10.1.12)* • Extended mandatory resources for devices (8.2.2) • Figure about transport protocol structure added to 9 • Unified and replaced the JSON keys “tag” and “POI” with “filter” in 9.1.1, 9.1.2, 9.2.2, 9.2.3 and whole chapter 10.1* • Added the JSON key “subResource” in 9.1.1, 9.1.2, 9.2.2, 9.2.3 and whole chapter 10.1* • Renamed the JSON key “CorrelationId” to correlationId” in 9.1.1, 9.1.2, 9.1.3, 9.2.1, 9.2.2 and whole chapter 10* • Reworked ServiceNetworkMessage (9.1.3, 9.2.16)* • Clarified usage of DataSetClassId in 9.2.1, 9.2.4 and 9.2.16 • Added the JSON key “context” to the resource config (9.3.3.1, 9.3.3.2, 10.1.3)* • Added definition of oi4_pv to data resource (9.3.7, 10.1.7) • Reworked event (9.3.9, 10.1.9)* ○ Renamed the JSON key “payload” to “details” inside the event resource (9.3.9, 10.1.9)* ○ Added the JSON key “origin“ to the event resource (9.3.9, 10.1.9)* ○ Renamed event category “opcSC” to “status” (8.1.6, 8.1.7, 9.3.9, 9.3.9.1 and 10.1.9)* ○ Renamed event category “undef” to “generic” (8.1.6, 8.1.7, 9.3.9, 9.3.9.4 and 10.1.9)* ○ Added sub-chapter 9.3.9.4 for generic events • Reworked publicationList (9.3.11, 10.1.11)* ○ Added “subResource” and changed “tag” to “filter”* ○ substitited “active” and “explicit” with “mode”*

Revision	Date	Author	Changes/comment
			○ Changed enumeration for “config”* • Added sub chapter 9.4 to explain arguments of Alliance defined methods. • Added/defined method “newDataSetWriterId” (9.4.5, 10.2.5) • Unified sequence diagrams in chapter 10 to follow the same color schema • Added event-mechanism to every “set” or “del” request, done in chapter 10.1* • Corrected typo “licText” to “licenseText” in chapter 10.1.5 as defined in 9.3.5 • Deleted method “findAssets” (former chapter 10.2.1) because this is already covered through other resources. • Deleted method “readPV” (former chapter 10.2.2) because this is already covered through other resources. • Deleted method “writePV” (former chapter 10.2.3) because this is already covered through other resources. • Actualized port information for OEC Registry (A1). • Actualized Glossary (A3) • Actualized list of Authors (A4) • Actualized appendix B1.1 • Actualized appendix B1.2 *) Breaking changes to Guideline V0.12

Table 18 Document history

Appendix B

B1 Examples for Message Bus communication

For all the following examples in B1.1 and B1.2 it is assumed that:

- The publisher of the information that uses the `pub` method is always the same. It uses the following serviceType (8.1.2) and oi4Identifier (3.1):
  - serviceType: `OTConnector`
  - oi4Identifier: `provider.com/FieldDataService/FDS-001/1200-0345`
- The requester of the information who uses the `get` method for this is the same who triggers changes by means of the `set` or `del` methods. It uses the following serviceType (8.1.2) and oi4Identifier (3.1):
  - serviceType: `ITConnector`
  - oi4Identifier: `consumer.com/MEService/MES%2fOnPrem/Inst07322`

B1.1 Resources

All following examples are related to chapter 10.1.

B1.1.1 mam

Get mam of an explicit asset

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/get/mam/provider.com/FieldDataService/FDS-001/1200-0345
```

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "",
  "Messages": []
}
```

Get mam of all assets related to this application

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/get/mam
```

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "Messages": []
}
```


Pub mam of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/mam/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was requested by previous published* `...get/mam/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

NOTE *The* `filter` *is either empty or omitted, because* `mam` *doesn't have any filter options.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 1,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 4,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": "Provider Ltd."
        },
        "ManufacturerUri": "provider.com",
        "Model": {
          "locale": "en-US",
          "text": "FieldDataService"
        },
        "ProductCode": "FDS-001",
        "HardwareRevision": "",
        "SoftwareRevision": "1.0.1",
        "DeviceRevision": "",
        "DeviceManual": "https://provider.com/manuals/FDS-001.pdf",
        "DeviceClass": "OI4.OTConnector",
        "SerialNumber": "1200-0345",
        "ProductInstanceUri": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "RevisionCounter": 1,
        "Description": {
          "locale": "en-US",
          "text": "This OT connector provides field data"
        }
      }
    }
  ]
}
```


Pub mam of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/mam`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

NOTE *For such static information, the optional* `Timestamp` *and maybe the optional* `SequenceNumber` *might not be needed.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 1,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": "Provider Ltd."
        },
        "ManufacturerUri": "provider.com",
        "Model": {
          "locale": "en-US",
          "text": "FieldDataService"
        },
        "ProductCode": "FDS-001",
        "HardwareRevision": "",
        "SoftwareRevision": "1.0.1",
        "DeviceRevision": "",
        "DeviceManual": "https://provider.com/manuals/FDS-001.pdf",
        "DeviceClass": "OI4.OTConnector",
        "SerialNumber": "1200-0345",
        "ProductInstanceUri": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "RevisionCounter": 1,
        "Description": {
          "locale": "en-US",
          "text": "This OT connector provides field data"
        }
      }
    },
    {
      //additional mam objects would be listed here if existing
    },
    {
      //in case too many mam objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with mam objects would be published
under the same topic
    }
  ]
}
```

Birth message of an application

`oi4/OTConnector/groe.mca/ServiceAModel/nd/fuli2/pub/mam/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *The timestamp in* `MessageId` *shall be set to 0. The optional* `correlationId` *is not relevant and is omitted or set to an empty string. The optional keys* `SequenceNumber` *and* `Timestamp` *shall not be used in this context.*

```
{
  "MessageId": "0000000000000-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "provider.com/FieldDataService/FDS-001/1200-0345",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "Messages": [
    {
      "DataSetWriterId": 1,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": "Provider Ltd."
        },
        "ManufacturerUri": "provider.com",
        "Model": {
          "locale": "en-US",
          "text": "FieldDataService"
        },
        "ProductCode": "FDS-001",
        "HardwareRevision": "",
        "SoftwareRevision": "1.0.1",
        "DeviceRevision": "",
        "DeviceManual": "https://provider.com/manuals/FDS-001.pdf",
        "DeviceClass": "OI4.OTConnector",
        "SerialNumber": "1200-0345",
        "ProductInstanceUri": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "RevisionCounter": 1,
        "Description": {
          "locale": "en-US",
          "text": "This OT connector provides field data"
        }
      }
    }
  ]
}
```


B1.1.2 health

Get health of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/health/provider.com/FieldDataService/FDS-001/1200-0345`


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "correlationId": "",
  "Messages": []
}
```


Get health of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/health`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "Messages": []
}
```


Pub health of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/health/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was requested by previous published* `...get/health/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

NOTE *The* `filter` *is either empty or omitted, because* `health` *doesn't have any filter options.*


```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 57,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "health": "NORMAL_0",
        "healthScore": 100
      }
    }
  ]
}
```


Pub health of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/health`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 58,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "health": "MAINTENANCE_REQUIRED_4",
        "healthScore": 55
      }
    },
    {
      //additional health objects would be listed here if existing
    },
    {
      //in case too many health objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with health objects would be published
under the same topic
    }
  ]
}
```

Disconnect message of an application

`oi4/OTConnector/groe.mca/ServiceAModel/nd/fuli2/pub/health/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *The timestamp in* `MessageId` *shall be set to 0. The optional* `correlationId` *is not relevant and is omitted or set to an empty string. The optional keys* `SequenceNumber` *and* `Timestamp` *shall not be used in this context.*

```
{
  "MessageId": "0000000000000-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "provider.com/FieldDataService/FDS-001/1200-0345",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "health": "NORMAL_0",
        "healthScore": 0
      }
    }
  ]
}
```

Will message of an application

`oi4/OTConnector/groe.mca/ServiceAModel/nd/fuli2/pub/health/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *The timestamp in* `MessageId` *shall be set to 0. The optional* `correlationId` *is not relevant and is omitted or set to an empty string. The optional keys* `SequenceNumber` *and* `Timestamp` *shall not be used in this context.*

```
{
  "MessageId": "0000000000000-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "provider.com/FieldDataService/FDS-001/1200-0345",
  "DataSetClassId": "d8e7b6df-42ba-448a-975a-199f59e8ffeb",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "health": "FAILURE_1",
        "healthScore": 0
      }
    }
  ]
}
```

B1.1.3 config

Get explicit config tag of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/config/provider.com/FieldDataService/FDS-001/1200-0345/Network%20settings`

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "",
  "Messages": []
}
```

Get all config tags of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/config/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *It is possible to request a specific locale setting as shown below. This might be useful for generic configuration dialogs, shown in different languages. In case the requested locale is not supported, the default* `"en-US"` *will be used.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 2,
      "filter": "<filter>",
      "subResource": "<oi4Identifier>",
      "Payload": {
        "locale": "de-DE"
      }
    }
  ]
}
```

Get config tags of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/config`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": []
}
```

Pub explicit config tag of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/config/provider.com/FieldDataService/FDS-001/1200-0345/Network%20settings`

NOTE *Let's assume, this publication was requested by previous published* `...get/config/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "network": {
          "name": {
            "locale": "en-US",
            "text": "Network"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes the network configuration of this service"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of Network DNS server"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": ""
          },
          "subnet-mask": {
            "description": {
              "locale": "en-US",
              "text": "Subnetmask of Network"
            },
            "name": {
              "locale": "en-US",
              "text": "Subnetmask"
            },
            "type": "String",
            "validation": {
              "pattern":
"^(((255\\.){3}(255|254|252|248|240|224|192|128|0+))|((255\\.){2}(255|254|
252|248|240|224|192|128|0+)\\.0)|((255\\.)(255|254|252|248|240|224|192|128
|0+)(\\.0+){2})|((255|254|252|248|240|224|192|128|0+)(\\.0+){3}))$"
            },
            "value": ""
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
```

```
              "text": "Port"
            },
            "type": "Number",
            "validation": {
              "min": 0,
              "max": 65535
            },
            "value": ""
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "Network settings"
          },
          "description": {
            "locale": "en-US",
            "text": "Settings of all network interfaces"
          }
        }
      }
    }
  ]
}
```


Pub all config tags of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/config/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "network": {
          "name": {
            "locale": "en-US",
            "text": "Network"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes the network configuration of this service"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of Network DNS server"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": ""
          },
          "subnet-mask": {
            "description": {
              "locale": "en-US",
              "text": "Subnetmask of Network"
            },
            "name": {
              "locale": "en-US",
              "text": "Subnetmask"
            },
            "type": "String",
            "validation": {
              "pattern":
"^(((255\\.){3}(255|254|252|248|240|224|192|128|0+))|((255\\.){2}(255|254|
252|248|240|224|192|128|0+)\\.0)|((255\\.)(255|254|252|248|240|224|192|128
|0+)(\\.0+){2})|((255|254|252|248|240|224|192|128|0+)(\\.0+){3}))$"
            },
            "value": ""
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Port"
```

```
            },
            "type": "Number",
            "validation": {
              "min": 0,
              "max": 65535
            },
            "value": ""
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "Network settings"
          },
          "description": {
            "locale": "en-US",
            "text": "Settings of all network interfaces"
          }
        }
      }
    },
    {
      //additional config objects would be listed here if existing
    },
    {
      //in case too many config objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with config objects would be published
under the same topic
    }
  ]
}
```

Pub config tags of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/config`

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "network": {
          "name": {
            "locale": "en-US",
            "text": "Network"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes the network configuration of this service"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of Network DNS server"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": ""
          },
          "subnet-mask": {
            "description": {
              "locale": "en-US",
              "text": "Subnetmask of Network"
            },
            "name": {
              "locale": "en-US",
              "text": "Subnetmask"
            },
            "type": "String",
            "validation": {
              "pattern":
"^(((255\\.){3}(255|254|252|248|240|224|192|128|0+))|((255\\.){2}(255|254|
252|248|240|224|192|128|0+)\\.0)|((255\\.)(255|254|252|248|240|224|192|128
|0+)(\\.0+){2})|((255|254|252|248|240|224|192|128|0+)(\\.0+){3}))$"
            },
            "value": ""
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Port"
            },
```

```
            "type": "Number",
            "validation": {
              "min": 0,
              "max": 65535
            },
            "value": ""
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "Network settings"
          },
          "description": {
            "locale": "en-US",
            "text": "Settings of all network interfaces"
          }
        }
      }
    },
    {
      //additional config objects would be listed here if existing
    },
    {
      //in case too many config objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with config objects would be published
under the same topic
    }
  ]
}
```

Set explicit config tag of an explicit asset (set new value)

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/set/config/provider.com/FieldDataService/FDS-001/1200-0345/Network%20settings`

NOTE *Not every DataSet is writable over the Message Bus. Check subscritionList and/or publicationList to find out about accessibility.*

NOTE *The payload must contain the values of mandatory config names. In addition, it may also contain the other properties received through* `pub/config`*. For details on the* `set` *format see section* [9.3.3.2]

Minimal set of values:

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "network": {
          "ip_address": {
            "value": "192.168.1.1"
          },
          "subnet-mask": {
            "value": "255.255.255.0"
          },
          "port": {
            "value": "80"
          }
        }
      }
    }
  ]
}
```


Full set of values:

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "network": {
          "name": {
            "locale": "en-US",
            "text": "Network"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes the network configuration of this service"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of Network DNS server"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": "192.168.1.1"
          },
          "subnet-mask": {
            "description": {
              "locale": "en-US",
              "text": "Subnetmask of Network"
            },
            "name": {
              "locale": "en-US",
              "text": "Subnetmask"
            },
            "type": "String",
            "validation": {
              "pattern":
"^(((255\\.){3}(255|254|252|248|240|224|192|128|0+))|((255\\.){2}(255|254|
252|248|240|224|192|128|0+)\\.0)|((255\\.)(255|254|252|248|240|224|192|128
|0+)(\\.0+){2})|((255|254|252|248|240|224|192|128|0+)(\\.0+){3}))$"
            },
            "value": "255.255.255.0"
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Port"
            },
```

```
            "type": "Number",
            "validation": {
              "min": 0,
              "max": 65535
            },
            "value": "80"
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "Network settings"
          },
          "description": {
            "locale": "en-US",
            "text": "Settings of all network interfaces"
          }
        }
      }
    }
  ]
}
```


Set explicit config tag of an explicit asset (empty existing values)

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/set/config/provider.com/FieldDataService/FDS-001/1200-0345/Network%20settings`

Minimal set of values:


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "network": {
          "ip_address": {
            "value": ""
          },
          "subnet-mask": {
            "value": ""
          },
          "port": {
            "value": ""
          }
        }
      }
    }
  ]
}
```

Full set of values:

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Network%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "network": {
          "name": {
            "locale": "en-US",
            "text": "Network"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes the network configuration of this service"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of Network DNS server"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": ""
          },
          "subnet-mask": {
            "description": {
              "locale": "en-US",
              "text": "Subnetmask of Network"
            },
            "name": {
              "locale": "en-US",
              "text": "Subnetmask"
            },
            "type": "String",
            "validation": {
              "pattern":
"^(((255\\.){3}(255|254|252|248|240|224|192|128|0+))|((255\\.){2}(255|254|
252|248|240|224|192|128|0+)\\.0)|((255\\.)(255|254|252|248|240|224|192|128
|0+)(\\.0+){2})|((255|254|252|248|240|224|192|128|0+)(\\.0+){3}))$"
            },
            "value": ""
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Port"
            },
```

```
              "type": "Number",
              "validation": {
                "min": 0,
                "max": 65535
              },
              "value": ""
            }
          },
          "context": {
            "name": {
              "locale": "en-US",
              "text": "Network settings"
            },
            "description": {
              "locale": "en-US",
              "text": "Settings of all network interfaces"
            }
          }
        }
      }
    ]
  }
```


Status event follows a set request

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/pub/event/status/OTConnector%2Fprovider.com%2FFieldDataService%2F
FDS-001%2F1200-0345
```

NOTE *Every* set *request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related set request.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "good",
      "subResource": "status",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 0,
        "description": "The operation succeeded.",
        "category": "CAT_STATUS_1"
        "details": {
          "symbolicId": "Good"
        }
      }
    }
  ]
}
```


More complex example with two groups

The "device1" group defines a set of configuration objects that must be set in a single call. Failure to do so may result in an invalid configuration of the service.

The "common" group is defined for everything that does not need to be configured together with other settings at the same time.

NOTE *Group what belongs together and should not be configured out of the context of other keys within this group.*

NOTE *In case a generic configuration tool is used, a group can always be displayed as one configuration dialog to make relationships clear.*

Get explicit config tag of an explicit asset

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/get/config/provider.com/FieldDataService/FDS-001/1200-
0345/Overall%20device%20settings
```

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "",
  "Messages": []
}
```


Pub explicit config tag of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/config/provider.com/FieldDataService/FDS-001/1200-0345/Overall%20device%20settings`

NOTE *Let's assume, this publication was requested by previous published* `...get/config/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Overall%20device%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "device1": {
          "name": {
            "locale": "en-US",
            "text": "Device 1"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes device one configuration options"
          },
          "aut0-r3fr3sh": {
            "mandatory": true,
            "name": {
              "locale": "en-US",
              "text": "Automatic Refresh Enabled"
            },
            "type": "Boolean",
            "value": "false"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of device to read values from"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?).){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": ""
          },
          "password": {
            "name": {
              "locale": "en-US",
              "text": "Password"
            },
            "sensitive": true,
            "type": "String",
            "value": ""
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
```

```
              "text": "Port"
            },
            "type": "Number",
            "validation": {
              "min": 0,
              "max": 65535
            },
            "value": ""
          },
          "protocol": {
            "mandatory": true,
            "name": {
              "locale": "en-US",
              "text": "Protocol"
            },
            "type": "String",
            "value": ""
          },
          "readInterval": {
            "description": {
              "locale": "en-US",
              "text": "Interval in which value is read from device"
            },
            "name": {
              "locale": "en-US",
              "text": "Read Interval"
            },
            "type": "Number",
            "value": ""
          },
          "unit": {
            "defaultValue": "°C",
            "description": {
              "locale": "en-US",
              "text": "Unit of measurement"
            },
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Unit"
            },
            "type": "String",
            "unit": "Temperature",
            "validation": {
              "values": ["K", "°C", "°De", "°F", "°N", "°R", "°Ré", "°Rø"
]
            },
            "value": ""
          },
          "user-name": {
            "name": {
              "locale": "en-US",
              "text": "Username"
            },
            "type": "String",
            "value": ""
          }
        },
        "common": {
          "name": {
            "locale": "en-US",
            "text": "general configuration"
          },
          "date-time": {
```

```
              "description": {
                "locale": "en-US",
                "text": "Date Time for service"
              },
              "name": {
                "locale": "en-US",
                "text": "Date Time"
              },
              "type": "DateTime",
              "value": ""
            }
          },
          "context": {
            "name": {
              "locale": "en-US",
              "text": "Overall device settings"
            },
            "description": {
              "locale": "en-US",
              "text": "Common and specific settings"
            }
          }
        }
      }
    ]
  }
```


Set explicit config tag of an explicit asset (set new value)

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/set/config/provider.com/FieldDataService/FDS-001/1200-0345/Overall%20device%20settings`

Minimal set of values:

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Overall%20device%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "device1": {
          "aut0-r3fr3sh": {
            "value": "false"
          },
          "ip_address": {
            "value": "192.168.1.123"
          },
          "password": {
            "value": "ssh-it's a secret!"
          },
          "port": {
            "value": "443"
          },
          "protocol": {
            "value": "HTTPS"
          },
          "readInterval": {
            "value": "100"
          },
          "unit": {
            "value": "K"
          },
          "user-name": {
            "value": "admin"
          }
        },
        "common": {
          "date-time": {
            "value": "2020-12-24T20:15:00Z"
          }
        }
      }
    }
  ]
}
```


Full set of values:

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Overall%20device%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "device1": {
          "name": {
            "locale": "en-US",
            "text": "Device 1"
          },
          "description": {
            "locale": "en-US",
            "text": "Describes device one configuration options"
          },
          "aut0-r3fr3sh": {
            "mandatory": true,
            "name": {
              "locale": "en-US",
              "text": "Automatic Refresh Enabled"
            },
            "type": "Boolean",
            "value": "false"
          },
          "ip_address": {
            "description": {
              "locale": "en-US",
              "text": "IP address of device to read values from"
            },
            "name": {
              "locale": "en-US",
              "text": "IP Address"
            },
            "type": "String",
            "validation": {
              "pattern": "^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?).){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
            },
            "value": "192.168.1.123"
          },
          "password": {
            "name": {
              "locale": "en-US",
              "text": "Password"
            },
            "sensitive": true,
            "type": "String",
            "value": "ssh-it's a secret!"
          },
          "port": {
            "defaultValue": "80",
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Port"
            },
```

```
            "type": "Number",
            "validation": {
              "min": 0,
              "max": 65535
            },
            "value": "443"
          },
          "protocol": {
            "mandatory": true,
            "name": {
              "locale": "en-US",
              "text": "Protocol"
            },
            "type": "String",
            "value": "HTTPS"
          },
          "readInterval": {
            "description": {
              "locale": "en-US",
              "text": "Interval in which value is read from device"
            },
            "name": {
              "locale": "en-US",
              "text": "Read Interval"
            },
            "type": "Number",
            "value": "100"
          },
          "unit": {
            "defaultValue": "°C",
            "description": {
              "locale": "en-US",
              "text": "Unit of measurement"
            },
            "mandatory": false,
            "name": {
              "locale": "en-US",
              "text": "Unit"
            },
            "type": "String",
            "unit": "Temperature",
            "validation": {
              "values": ["K", "°C", "°De", "°F", "°N", "°R", "°Ré", "°Rø"
]
            },
            "value": "K"
          },
          "user-name": {
            "name": {
              "locale": "en-US",
              "text": "Username"
            },
            "type": "String",
            "value": "admin"
          }
        },
        "common": {
          "name": {
            "locale": "en-US",
            "text": "general configuration"
          },
          "date-time": {
            "description": {
              "locale": "en-US",
```

```
              "text": "Date Time for service"
            },
            "name": {
              "locale": "en-US",
              "text": "Date Time"
            },
            "type": "DateTime",
            "value": "2020-12-24T20:15:00Z"
          }
        },
        "context": {
          "name": {
            "locale": "en-US",
            "text": "Overall device settings"
          },
          "description": {
            "locale": "en-US",
            "text": "Common and specific settings"
          }
        }
      }
    }
  ]
}
```


Set explicit config tag of an explicit asset (empty existing values)

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/set/config/provider.com/FieldDataService/FDS-001/1200-0345/Overall%20device%20settings`

Minimal set of values:

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "9d5983db-440d-4474-9fd7-1cd7a6c8b6c2",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "filter": "Overall%20device%20settings",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "device1": {
          "aut0-r3fr3sh": {
            "value": ""
          },
          "ip_address": {
            "value": ""
          },
          "password": {
            "value": ""
          },
          "port": {
            "value": ""
          },
          "protocol": {
            "value": ""
          },
          "readInterval": {
            "value": ""
          },
          "unit": {
            "value": ""
          },
          "user-name": {
            "value": "admin"
          }
        },
        "common": {
          "date-time": {
            "value": ""
          }
        }
      }
    }
  ]
}
```


Status event follows a set request

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/event/status/OTConnector%2Fprovider.com%2FFieldDataService%2FFDS-001%2F1200-0345`

NOTE *Every* set *request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related set request.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "good",
      "subResource": "status",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 0,
        "description": "The operation succeeded.",
        "category": "CAT_STATUS_1"
        "details": {
          "symbolicId": "Good"
        }
      }
    }
  ]
}
```


B1.1.4 license

Get an explicit license of an explicit asset

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/get/license/provider.com/FieldDataService/FDS-001/1200-0345/MIT
```


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "",
  "Messages": []
}
```


Get all licenses of an explicit asset

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/get/license/provider.com/FieldDataService/FDS-001/1200-0345
```

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "",
  "Messages": []
}
```

Get all licenses of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/license`

NOTE *The optional* `correlationId` *is not needed, if request was not triggered from another service.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "Messages": []
}
```

Pub an explicit license of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/license/provider.com/FieldDataService/FDS-001/1200-0345/MIT`

NOTE *Let's assume, this publication was requested by previous published* `...get/license/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 357,
      "filter": "MIT",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 21,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "components": [
          {
            "component": "MQTTCl",
            "licAuthors": [
              "Lorem ipsum"
            ],
            "licAddText": "Copyright (c) 2020 Lorem ipsum"
          },
          {
            "component": "nodeFDS",
            "licAuthors": [
              "provider.com"
            ],
            "licAddText": "Copyright (c) 2021 provider.com"
          }
        ]
      }
    }
  ]
}
```


Pub all licenses of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/license/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was requested by previous published* `...get/license/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

NOTE *For such static information, the optional* `Timestamp` *and maybe the optional* `SequenceNumber` *might not be needed.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 357,
      "filter": "MIT",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "components": [
          {
            "component": "MQTTCl",
            "licAuthors": [
              "Lorem ipsum"
            ],
            "licAddText": "Copyright (c) 2020 Lorem ipsum"
          },
          {
            "component": "nodeFDS",
            "licAuthors": [
              "provider.com"
            ],
            "licAddText": "Copyright (c) 2021 provider.com"
          }
        ]
      }
    },
    {
      "DataSetWriterId": 358,
      "filter": "Apache%202.0",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "components": [
          {
            "component": "nodeFDS_field",
            "licAuthors": [
              "provider.com"
            ],
            "licAddText": "Copyright 2021 provider.com"
          }
        ]
      }
    },
    {
      //additional license objects would be listed here if existing
    },
    {
      //in case too many license objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with license objects would be
published under the same topic
    }
  ]
}
```


Pub all licenses of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/license`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 357,
      "filter": "MIT",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "components": [
          {
            "component": "MQTTCl",
            "licAuthors": [
              "Lorem ipsum"
            ],
            "licAddText": "Copyright (c) 2020 Lorem ipsum"
          },
          {
            "component": "nodeFDS",
            "licAuthors": [
              "provider.com"
            ],
            "licAddText": "Copyright (c) 2021 provider.com"
          }
        ]
      }
    },
    {
      "DataSetWriterId": 358,
      "filter": "Apache%202.0",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "components": [
          {
            "component": "nodeFDS_field",
            "licAuthors": [
              "provider.com"
            ],
            "licAddText": "Copyright 2021 provider.com"
          }
        ]
      }
    },
    {
      //additional license objects would be listed here if existing
    },
    {
      //in case too many license objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with license objects would be
published under the same topic
    }
  ]
}
```


B1.1.5 licenseText

Get an explicit licenseText of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/licenseText/provider.com/FieldDataService/FDS-001/1200-0345/MIT`


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "",
  "Messages": []
}
```


Get all licenseTexts of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/licenseText/provider.com/FieldDataService/FDS-001/1200-0345`


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "",
  "Messages": []
}
```


Get all licenseTexts of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/licenseText`

NOTE *The optional* `correlationId` *is not needed, if request was not triggered from another service.*


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "Messages": []
}
```


Pub an explicit licenseText of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/licenseText/provider.com/FieldDataService/FDS-001/1200-0345/MIT`

NOTE *Let's assume, this publication was requested by previous published* `...get/licenseText/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 489,
      "filter": "MIT",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 17,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "licenseText": "Permission is hereby granted, free of charge, to
any person obtaining a copy of this software and associated documentation
files (the \"Software\"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and\/or sell copies of the Software, and
to permit persons to whom the Software is furnished to do so, subject to
the following conditions:\nThe above copyright notice and this permission
notice shall be included in all copies or substantial portions of the
Software.\nTHE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
USE OR OTHER DEALINGS IN THE SOFTWARE."
      }
    }
  ]
}
```


Pub all licenseTexts of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/licenseText/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was requested by previous published* `...get/licenseText/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

NOTE *For such static information, a* `Timestamp` *and maybe* `SequenceNumber` *might not be needed.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 489,
      "filter": "MIT",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "licenseText": "Permission is hereby granted, free of charge, to
any person obtaining a copy of this software and associated documentation
files (the \"Software\"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and\/or sell copies of the Software, and
to permit persons to whom the Software is furnished to do so, subject to
the following conditions:\nThe above copyright notice and this permission
notice shall be included in all copies or substantial portions of the
Software.\nTHE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
USE OR OTHER DEALINGS IN THE SOFTWARE."
      }
    },
    {
      "DataSetWriterId": 490,
      "filter": "Apache%202.0",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "licenseText": "Licensed under the Apache License, Version 2.0
(the \"License\");\nyou may not use this file except in compliance with
the License.\nYou may obtain a copy of the License at\n
http:\/\/www.apache.org\/licenses\/LICENSE-2.0\nUnless required by
applicable law or agreed to in writing, software distributed under the
License is distributed on an \"AS IS\" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.\nSee the License for
the specific language governing permissions and limitations under the
License."
      }
    },
    {
      //additional licenseText objects would be listed here if existing
    },
    {
      //in case too many licenseText objacts are existing, a pagination
object would point that out
      //and further NetworkMessages with licenseText objects would be
published under the same topic
    }
  ]
}
```


Pub all licenseTexts of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/licenseText`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "a6e6c727-4057-419f-b2ea-3fe9173e71cf",
  "correlationId": "",
  "Messages": [
   {
      "DataSetWriterId": 489,
      "filter": "MIT",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "licenseText": "Permission is hereby granted, free of charge, to
any person obtaining a copy of this software and associated documentation
files (the \"Software\"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and\/or sell copies of the Software, and
to permit persons to whom the Software is furnished to do so, subject to
the following conditions:\nThe above copyright notice and this permission
notice shall be included in all copies or substantial portions of the
Software.\nTHE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
USE OR OTHER DEALINGS IN THE SOFTWARE."
      }
    },
    {
      "DataSetWriterId": 490,
      "filter": "Apache%202.0",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "licenseText": "Licensed under the Apache License, Version 2.0
(the \"License\");\nyou may not use this file except in compliance with
the License.\nYou may obtain a copy of the License at\n
http:\/\/www.apache.org\/licenses\/LICENSE-2.0\nUnless required by
applicable law or agreed to in writing, software distributed under the
License is distributed on an \"AS IS\" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.\nSee the License for
the specific language governing permissions and limitations under the
License."
      }
    },
    {
      //additional licenseText objects would be listed here if existing
    },
    {
      //in case too many licenseText objacts are existing, a pagination
object would point that out
      //and further NetworkMessages with licenseText objects would be
published under the same topic
    }
  ]
}
```


B1.1.6 rtLicense

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

B1.1.7 data

Get explicit data tag of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/data/provider.com/FieldDataService/FDS-001/1200-0345/oee`

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "",
  "correlationId": "",
  "Messages": []
}
```

Get all data tags of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/data/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Most data tags will not have a* `DataSetClassId`*, because they are not globally standardized such as mam, health and other submodels in context of the Alliance. Therefore the* `DataSetClassId` *is either empty or omitted as in this example.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "correlationId": "",
  "Messages": []
}
```

Get data tags of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/data`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": []
}
```

Pub explicit data tag of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/data/provider.com/FieldDataService/FDS-001/1200-0345/oee`

NOTE *Let's assume, this publication was requested by previous published* `...get/data/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*


```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "filter": "oee",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 3985,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "availability": 90,
        "performanceRate": 95,
        "qualityRate": 98,
        "product": 84
      }
    }
  ]
}
```


Pub all data tags of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/data/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "filter": "oee",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 3985,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "availability": 90,
        "performanceRate": 95,
        "qualityRate": 98,
        "product": 84
      }
    },
    {
      //additional data objects would be listed here if existing
    },
    {
      //in case too many data objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with data objects would be published
under the same topic
    }
  ]
}
```

Pub data tags of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/data`

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "filter": "oee",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 3985,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "availability": 90,
        "performanceRate": 95,
        "qualityRate": 98,
        "product": 84
      }
    },
    {
      //additional mam objects would be listed here if existing
    },
    {
      //in case too many mam objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with mam objects would be published
under the same topic
    }
  ]
}
```

Set explicit data tag of an explicit asset

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/set/data/provider.com/FieldDataService/FDS-001/1200-0345/oee
```

NOTE *Not every DataSet is writable over the Message Bus. Check subscritionList and/or publicationList to find out about accessibility.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
     {
      "DataSetWriterId": 93,
      "filter": "oee",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 3985,
      "Timestamp": "2021-12-16T22:25:10.000Z",
      "Payload": {
        "availability": 99,
        "performanceRate": 99,
        "qualityRate": 99,
        "product": 97
      }
    }
  ]
}
```


Status event follows a set request

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/pub/event/status/OTConnector%2Fprovider.com%2FFieldDataService%2F
FDS-001%2F1200-0345
```

NOTE *Every* set *request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related set request.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "good",
      "subResource": "status",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 0,
        "description": "The operation succeeded.",
        "category": "CAT_STATUS_1"
        "details": {
          "symbolicId": "Good"
        }
      }
    }
  ]
}
```


B1.1.8 metadata

Get metadata of an explicit data tag

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/metadata/provider.com/FieldDataService/FDS-001/1200-0345/oee`


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-metadata",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetWriterId": 753,
  "filter": "oee",
  "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
  "MetaData": {}
}
```


Pub metadata of an explicit data tag

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/metadata/provider.com/FieldDataService/FDS-001/1200-0345/<tag>`

NOTE *Let's assume, this publication was requested by previous published* `...get/metadata/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

```
{
  "MessageId": "1639693510094-provider.com/FieldDataService/FDS-001/1200-
0345",
  "MessageType": "ua-metadata",
  "PublisherId": "provider.com/FieldDataService/FDS-001/1200-0345",
  "DataSetWriterId": 753,
  "filter": "oee",
  "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MetaData": {
    <DataSetMetaDataType>
  }
}
```

Set metadata of an explicit data tag

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/set/metadata/provider.com/FieldDataService/FDS-001/1200-0345/<tag>`

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-metadata",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetWriterId": 753,
  "filter": "oee",
  "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
  "MetaData": {
    <DataSetMetaDataType>
  }
}
```

Status event follows a set request

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/event/status/OTConnector%2Fprovider.com%2FFieldDataService%2FFDS-001%2F1200-0345`

NOTE *Every* set *request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related set request.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "good",
      "subResource": "status",
      "SequenceNumber": 38,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 0,
        "description": "The operation succeeded.",
        "category": "CAT_STATUS_1"
        "details": {
          "symbolicId": "Good"
        }
      }
    }
  ]
}
```


B1.1.9 event

NOTE *Only DataSetMessages of type* `event` *with same* `category` *and* `eventFilter` *can be combined in a single NetworkMessage.*

Pub event of category status (OPC UA Statuscode)

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/event/status/good`

NOTE *Let's assume, this event is a reaction of a previous made publication on the Message Bus - the* `correlationId` *points to the* `MessageId`*, which triggered this event.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "good",
      "subResource": "status",
      "SequenceNumber": 14,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 0,
        "description": "The operation succeeded.",
        "category": "CAT_STATUS_1"
        "details": {
          "symbolicId": "Good"
        }
      }
    }
  ]
}
```


Pub event of category syslog

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/event/syslog/notice`

NOTE *Let's assume, this event was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*

NOTE *The optional key* `description` *is not used in context of category* `syslog`*.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "Messages": [
    {
      "DataSetWriterId": 625,
      "filter": "notice",
      "subResource": "syslog",
      "SequenceNumber": 15,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": <RPI>,
        "category": "CAT_SYSLOG_0"
        "details": {
          "MSG": "<MSG>",
          "HEADER": "<HEADER>"
        }
      }
    }
  ]
}
```


Pub event of category ne107 (NAMUR NE107)

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/event/ne107/maintenanceRequired`


```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "maintenanceRequired",
      "subResource": "ne107",
      "SequenceNumber": 16,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 4,
        "description": "Maintenance Required",
        "category": "CAT_NE107_2"
        "details": {
          "diagnosticCode": "F-238",
          "location": ""
        }
      }
    }
  ]
}
```


Pub event of category generic

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/event/generic/high`

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "Messages": [
    {
      "DataSetWriterId": 625,
      "filter": "high",
      "subResource": "generic",
      "SequenceNumber": 17,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": <application defined number>,
        "description": "<application defined description>",
        "category": "CAT_GENERIC_99"
        "details": {
          <application defined opject>
        }
      }
    }
  ]
}
```

B1.1.10 profile

Get profile of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/profile/provider.com/FieldDataService/FDS-001/1200-0345`

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "48017c6a-05c8-48d7-9d85-4b08bbb707f3",
  "correlationId": "",
  "Messages": []
}
```

Get profiles of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/profile`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "48017c6a-05c8-48d7-9d85-4b08bbb707f3",
  "Messages": []
}
```


Pub profile of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/profile/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was requested by previous published* `...get/profile/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication*

NOTE *The* `filter` *is either empty or omitted, because* `profile` *doesn't have any filter options.*


```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "48017c6a-05c8-48d7-9d85-4b08bbb707f3",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 74,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 42,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "resource": [
          "mam",
          "health",
          "license",
          "licenseText",
          "rtLicense",
          "profile",
          "publicationList",
          "subscriptionList"
        ]
      }
    }
  ]
}
```


Pub profiles of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/profile`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example*

NOTE *For such static information, a* `Timestamp` *and maybe* `SequenceNumber` *might not be needed.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "48017c6a-05c8-48d7-9d85-4b08bbb707f3",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 74,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": [
          "mam",
          "health",
          "license",
          "licenseText",
          "rtLicense",
          "profile",
          "publicationList",
          "subscriptionList"
        ]
      }
    },
    {
      //additional profile objects would be listed here if existing
    },
    {
      //in case too many profile objacts are existing, a pagination object
would point that out
      //and further NetworkMessages with profile objects would be
published under the same topic
    }
  ]
}
```

B1.1.11 publicationList

NOTE *Depending on which resource a searched* `publicationList` *entry belongs to, different topics are needed to filter it out. The following construct can be used to filter out each publicationList entry:*
`oi4/<serviceType>/<appId>/<method>/publicationList/[<oi4Identifier>[/<resourceType>[/<tag>]]]`

NOTE *For a* `publicationList` *entry belonging to the resource* `data`*, the filter is a combination of* `resourceType` *and* `tag`*.*

NOTE *For a* `publicationList` *entry belonging to the resource* `mam`*, the filter consists only of* `resourceType`*.*

Get an explicit publicationList entry

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/publicationList/provider.com/FieldDataService/FDS-001/1200-0345/data/oee`

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "",
  "Messages": []
}
```


Get all publicationList entries of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/publicationList/provider.com/FieldDataService/FDS-001/1200-0345`


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "",
  "Messages": []
}
```


Get full publicationList of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/publicationList`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*


```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "Messages": []
}
```


Pub an explicit publicationList entry

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/publicationList/provider.com/FieldDataService/FDS-001/1200-0345/data/oee`

NOTE *Let's assume, this publication was requested by previous published* `...get/publicationList/...` *- the* `correlationId` *points to the* `MessageId`*, which requested this publication.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 161,
      "filter": "data",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": "data",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "filter": "oee",
        "DataSetWriterId": 93,
        "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "config": "MODE_AND_INTERVAL_3"
      }
    }
  ]
}
```


Pub all publicationList entries of an explicit asset

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/publicationList/provider.com/FieldDataService/FDS-001/1200-0345`

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either omitted or empty as in this example.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "correlationId": "",
  "Messages": [
    {
      "DataSetWriterId": 161,
      "filter": "mam",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": "mam",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 1,
        "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "filter": "health",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": "health",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 3456,
        "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "interval": 0,
        "config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "filter": "profile",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
          "resource": "profile",
          "subResource": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "DataSetWriterId": 74,
          "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
          "interval": 0,
          "config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "filter": "event",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
          "resource": "event",
          "subResource": "CAT_SYSLOG_0",
          "filter": "notice",
          "DataSetWriterId": 625,
```

```
          "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "mode": "APPLICATION_2",
          "interval": 0,
          "config": "NONE_0"
        }
      },
      {
        "DataSetWriterId": 161,
        "filter": "data",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "Payload": {
          "resource": "data",
          "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
          "filter": "oee",
          "DataSetWriterId": 93,
          "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
          "config": "MODE_AND_INTERVAL_3"
        }
      },
      {
        //additional publicationList objects would be listed here if
existing
      },
      {
        //in case too many publicationList objacts are existing, a
pagination object would point that out
        //and further NetworkMessages with publicationList objects would be
published under the same topic
      }
    ]
}
```

Pub full publicationList of all assets related to this application

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/publicationList`

NOTE *For such static information, the optional* `Timestamp` *and the optional* `SequenceNumber` *might not be needed. Several other keys, such as* `precisions` *and* `interval` *are having default behavior, if not present.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "Messages": [
     {
      "DataSetWriterId": 161,
      "filter": "mam",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": "mam",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 1,
        "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "filter": "health",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": "health",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 3456,
        "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "interval": 0,
        "config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "filter": "profile",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
          "resource": "profile",
          "subResource": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "DataSetWriterId": 74,
          "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
          "config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "filter": "event",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
          "resource": "event",
          "subResource": "CAT_SYSLOG_0",
          "filter": "notice",
          "DataSetWriterId": 625,
          "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "mode": "APPLICATION_2",
```

```
            "config": "NONE_0"
        }
      },
      {
        "DataSetWriterId": 161,
        "filter": "data",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "Payload": {
          "resource": "data",
          "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
          "filter": "oee",
          "DataSetWriterId": 93,
          "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
          "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
          "config": "MODE_AND_INTERVAL_3"
        }
      },
      {
        //additional publicationList objects would be listed here if
existing
      },
      {
        //in case too many publicationList objacts are existing, a
pagination object would point that out
        //and further NetworkMessages with publicationList objects would be
published under the same topic
      }
    ]
}
```

Set an explicit publicationList entry

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/set/publicationList/provider.com/FieldDataService/FDS-001/1200-0345/Network%20settings`

Let's assume, we want to change the publication behavior from "on change" to "cyclic" to get actualized data every 1000 ms.

NOTE *If the key config is unequal to NONE_0, the publicationList entry is editable over the Message Bus.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "Messages": [
    {
      "DataSetWriterId": 161,
      "filter": "data",
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "resource": "data",
        "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
        "filter": "oee",
        "DataSetWriterId": 93,
        "oi4Identifier": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "interval": 1000,
        "config": "MODE_AND_INTERVAL_3"
      }
    }
  ]
}
```


Status event follows a set request

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/pub/event/status/OTConnector%2Fprovider.com%2FFieldDataService%2F
FDS-001%2F1200-0345
```

NOTE *Every* set *request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related set request.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "filter": "good",
      "subResource": "status",
      "SequenceNumber": 39,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "origin": "provider.com/FieldDataService/FDS-001/1200-0345",
        "number": 0,
        "description": "The operation succeeded.",
        "category": "CAT_STATUS_1"
        "details": {
          "symbolicId": "Good"
        }
      }
    }
  ]
}
```


Excursion on how to use precision:

dataSet (example)	precision (full array for dataSet example)	precision (shorted array for dataSet example)
Assume, we have a dataSet, representing some motor behavior. We want to publish only use case relevant changes inside this dataSet, but some of the values inside the dataSet are quite “floating”:	We want to reduce unnecessary traffic on the Message Bus and we are using the precision functionality to do so. Two out of four values inside the given dataSet should be set up to use precision functionality - “temperature” and “speed”:	We can reduce the number of precisionObjects, when we only list the values, which shall use other than default (on change) precision:
ATTENTION: Only the dataSet, not the whole NetworkMessage is listed here!	**ATTENTION:** Only the precision part, not the whole publicationList entry! **ATTENTION:** This is the full/detailed precision:	**ATTENTION:** Only the precision part, not the whole publicationList entry! **ATTENTION:** This is the reduced/necessary precision:
`"Payload": { "temperature": 37.2, "speed": 1500, "torque": 44, "direction": "left" }`	`"precisions": { "temperature": 0.5, //next pub would be on <=36.7 or >=37.7 "speed": 10, //next pub would be on <=1490 or >=1510 "torque": 0 //default, publish on change }`	`"precisions": { "temperature": 0.5, //next pub would be on <=36.7 or >=37.7 "speed": 10 //next pub would be on <=1490 or >=1510 }`

dataSet (example)	precision (full array for dataSet example)	precision (shorted array for dataSet example)
	NOTE: The object for “direction” is missing in above's array, because its value is of type STRING. precision is only usable for values of number based data types.	**NOTE:** The object for “torque” is missing in above's array, because it should publish on change (precision = 0 = default).

B1.1.13 interface

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

B1.1.15 pagination

Get mam of all assets related to this application with pagination

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/mam`

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "Messages": [
    {
      "DataSetWriterId": 2,
      "Payload": {
        "perPage": 25,
        "page": 1
      }
    }
  ]
}
```

Pub mam of all assets related to this application with pagination

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/mam`

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 1,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": "Provider Ltd."
        },
        "ManufacturerUri": "provider.com",
        "Model": {
          "locale": "en-US",
          "text": "FieldDataService"
        },
        "ProductCode": "FDS-001",
        "HardwareRevision": "",
        "SoftwareRevision": "1.0.1",
        "DeviceRevision": "",
        "DeviceManual": "https://provider.com/manuals/FDS-001.pdf",
        "DeviceClass": "OI4.OTConnector",
        "SerialNumber": "1200-0345",
        "ProductInstanceUri": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "RevisionCounter": 1,
        "Description": {
          "locale": "en-US",
          "text": "This OT connector provides field data"
        }
      }
    },
    {
      "DataSetWriterId": 4798,
      "subResource": "groe.mca/DeviceModelB/ModelA/AC33.447/20443",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": "GROE"
        },
        "ManufacturerUri": "https://groe.mca",
        "Model": {
          "locale": "en-US",
          "text": "DeviceModelB"
        },
        "ProductCode": "AC33.447",
        "HardwareRevision": "1.1",
        "SoftwareRevision": "2.4.1",
        "DeviceRevision": "4",
        "DeviceManual": "",
        "DeviceClass": "",
        "SerialNumber": "20443",
        "ProductInstanceUri": "groe.mca/DeviceModelB/AC33.447/20443",
        "RevisionCounter": 1,
        "Description": {
          "locale": "en-US",
          "text": "Device Model B measurement unit"
```

```
          }
        }
      },
      {
        "DataSetWriterId": 2,
        "Payload": {
          "totalCount": 2
          "perPage": 25,
          "page": 1,
          "hasNext": false
        }
      }
    ]
  }
```

B1.1.16 locale

Get mam of all assets related to this application with locale

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/get/mam`

```
  {
    "MessageId": "1639693510000-
  ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
    "MessageType": "ua-data",
    "PublisherId":
  "ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
    "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
    "Messages": [
      {
        "DataSetWriterId": 3,
        "Payload": {
          "locale": "de-DE"
        }
      }
    ]
  }
```

Pub mam of all assets related to this application with locale

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/pub/mam`

NOTE *If the requested* `locale` *is not available or only partially available, the default "en-US" settings will be used in these cases.*

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "correlationId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 1,
      "subResource": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Manufacturer": {
          "locale": "de-DE",
          "text": "Provider GmbH"
        },
        "ManufacturerUri": "provider.com",
        "Model": {
          "locale": "de-DE",
          "text": "FieldDataService"
        },
        "ProductCode": "FDS-001",
        "HardwareRevision": "",
        "SoftwareRevision": "1.0.1",
        "DeviceRevision": "",
        "DeviceManual": "https://provider.com/manuals/FDS-001.pdf",
        "DeviceClass": "OI4.OTConnector",
        "SerialNumber": "1200-0345",
        "ProductInstanceUri": "provider.com/FieldDataService/FDS-001/1200-
0345",
        "RevisionCounter": 1,
        "Description": {
          "locale": "de-DE",
          "text": "Das ist ein OT Konnektor und stellt Felddaten zur
Verfügung"
        }
      }
    },
    {
      "DataSetWriterId": 4798,
      "subResource": "groe.mca/DeviceModelB/ModelA/AC33.447/20443",
      "Payload": {
        "Manufacturer": {
          "locale": "en-US",
          "text": "GROE"
        },
        "ManufacturerUri": "https://groe.mca",
        "Model": {
          "locale": "en-US",
          "text": "DeviceModelB"
        },
        "ProductCode": "AC33.447",
        "HardwareRevision": "1.1",
        "SoftwareRevision": "2.4.1",
        "DeviceRevision": "4",
        "DeviceManual": "",
        "DeviceClass": "",
        "SerialNumber": "20443",
        "ProductInstanceUri": "groe.mca/DeviceModelB/AC33.447/20443",
        "RevisionCounter": 1,
        "Description": {
          "locale": "en-US",
```

```
            "text": "Device Model B measurement unit"
          }
        }
      },
      {
        "DataSetWriterId": 2,
        "Payload": {
          "totalCount": 2
          "perPage": 25,
          "page": 1,
          "hasNext": false
        }
      }
    ]
  }
```


B1.2 Common services

B1.2.1 fileUpload

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

B1.2.2 fileDownload

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

B1.2.3 firmwareUpdate

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

B1.2.4 blink

ATTENTION *Information in this chapter requires alignment with other working groups and has not been maturely specified.*

B1.2.5 newDataSetWriterId

The ITConnector is able to provide information to generate the `referenceDesignation` of the OTConnector. To do so, the ITConnector need a valide DataSetWriterId fron the OTConnector, which it requests by calling the method `newDataSetWriterId`.

Call for newDataSetWriterId

```
oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/call/newDataSetWriterId
```

NOTE *Let's assume, this publication was self initialized. Therefore the* `correlationId` *is either empty or omitted as in this example.*

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "MessageType": "MSG",
  "PublisherId":
"ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "DataSetClassId": "2aca55bd-0d6f-41b1-a1c2-2d61afcc21f0",
  "Message": {
    "methodsToCall": [
      {
        "methodId": "newDataSetWriterId",
        "inputArguments": [
          {
            "resource": "referenceDesignation"
          }
        ]
      }
    ]
  }
}
```

Reply for newDataSetWriterId

`oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/reply/newDataSetWriterId`

NOTE *A reply always follows a call, therefore we always have to fill the* `correlationId` *with the* `MessageId` of the caller.

```
{
  "MessageId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "MSG",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345",
  "DataSetClassId ": "2aca55bd-0d6f-41b1-a1c2-2d61afcc21f0",
  "correlationId ": "1639693510000-
ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Message": {
    "results": [
      {
        "statusCode": 0,
        "inputArgmentResults": [0],
        "outputArguments": [
          {
            "DataSetWriterId": 57211,
            "ttl": 10
          }
        ]
      }
    ]
  }
}
```