



**OPEN INDUSTRY 4.0
ALLIANCE**

Guideline

Development Guideline for Open Edge Computing

IMPRINT

Publisher

Open Industry 4.0 Alliance

Christoph Merian-Ring 12, 4153 Reinach, Switzerland

<https://openindustry4.com>

info@openindustry4.com

Status

File: Development Guideline for Open Edge Computing

October 10th, 2023 – Version 1.1.1

Editors

Konrad Heidrich (Hilscher Gesellschaft für Systemautomation mbH)

Authors

Listed in [A 5](#)

Preface

The *Open Edge Computing (OEC) - Development Guideline* is a document which is the result of the activities within the Open Edge Computing work 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 within the Open Industry 4.0 Alliance requirements and architecture. As this document is a guideline and not a specification, some of the content is still ambiguous or in discussion as well as some of the definitions given may be subject to change.

This document primarily focuses on describing the message exchange between the field, edge and cloud layers within the overall architecture of the Open Industry 4.0 Alliance. The messages defined in the Open Edge Computing layer are based on existing industry standards such as OPC UA and MQTT. In the [second chapter](#), this document introduces the basic concepts, models and paradigms necessary to build the fundamentals of Open Edge Computing. Chapter [3](#) and [4](#) present two essential concepts of the Open Edge Computing layer: the [Oi4Identifier](#) and the [Master Asset Model \(MAM\)](#). Chapter [5](#) provides detailed processes and scenarios that describe the interactions of the components within the Edge Computing layer with the underlying layer (field devices and assets) as well as the overlying layer (cloud). While chapters [6](#) elaborates requirements and settings for application 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](#) elaborates on the Message Bus communications, building upon the definitions and concepts presented in the previous sections. Finally, chapter [11](#) introduces the requirements of the Open Edge Computing platform regarding components consisting within the architecture, hardware, applications, and software licenses.

The technical committee of the Open Industry 4.0 Alliance has achieved quick success in promoting and achieving interoperability. This is thanks to our open and result-oriented mindset, as well as the strong commitment of our founder partners. We warmly invite new members to join us in this collaborative spirit, as we work towards creating a practical and 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, former lead of the technical committee of the Open Industry 4.0 Alliance

Content

1	Conventions	10
2	Introduction to Open Edge Computing	
	- Considered Standards & Technical Decisions	12
2.1	Introduction to Edge Computing	13
2.2	Introduction to MQTT.....	14
2.2.1	Standardization	14
2.2.2	Communication Principles of MQTT - Pub/Sub Mechanism	15
2.3	Introduction to OPC UA PubSub	15
2.4	Introduction to Containerization	16
2.5	Cyber Security	17
2.5.1	IEC 62443	17
2.5.2	DIN SPEC 27070.....	18
3	The Oi4Identifier	20
3.1	Structure of the Oi4Identifier	20
3.2	Additional Provisions Regarding the Oi4Identifier	22
4	The Master Asset Model	24
5	Overall Process Description	29
5.1	Layer Interactions by Use Cases.....	29
5.1.1	Asset Onboarding	29
5.1.2	Condition Monitoring (Health)	30
5.1.3	Handling Process Data	31
5.2	Detailed Container Interactions Inside the Open Edge Computing Layer	33
5.2.1	Onboarding of a Field Device	33
5.2.2	Condition Monitoring (Health) of a Field Device.....	34
5.2.3	Process Data Ingestion of a Field Device	35
6	Container Environment	37

6.1	Container Storage.....	37
6.1.1	Message Bus Storage.....	37
6.1.2	OID Certificate Storage	39
6.1.3	Secret Storage.....	40
6.1.4	Application Specific Storages	42
6.2	Container Name Conventions.....	43
6.3	Container Image Integrity.....	44
6.4	Container Networks	45
7	General Requirements on MQTT.....	48
7.1	Protocol Version.....	48
7.2	Client Connection Setup.....	49
7.2.1	Broker Address and Ports	49
7.2.2	Security Settings.....	49
7.2.3	Client Identifier.....	50
7.2.4	Keep Alive	50
7.2.5	Clean Session	50
7.3	Client Message Setup.....	51
7.3.1	Quality of Service Level.....	51
7.3.2	Retained Messages	52
7.3.3	Birth Message	52
7.3.4	Close Message	54
7.3.5	Will Message	54
7.4	General Behavior of MQTT on Changes	56
8	Topics Definition.....	57
8.1	Topic Namespace Elements.....	57
8.1.1	Namespace Element.....	58
8.1.2	ServiceType Element	58
8.1.3	AppId Element.....	60
8.1.4	Method Element	60
8.1.5	Resource Element.....	61
8.1.5.1	Resources.....	62
8.1.5.2	Common Services	64
8.1.5.3	Specific Services.....	65
8.1.6	Source Element.....	66

8.1.7	Filter Element	66
8.2	Minimum Set of Topic Elements for Open Industry 4.0 Alliance Compliance	71
8.2.1	For Applications.....	71
8.2.2	For Devices	72
9	Payload Format	74
9.1	Structure of the Defined MessageTypes.....	76
9.1.1	Overview of the OPC UA Conforming MessageType ua-data.....	76
9.1.2	Overview of the OPC UA Conforming MessageType ua-metadata	77
9.1.3	General MessageType MSG in Accordance with OPC UA	78
9.2	OPC UA Objects - Defined by OPC Foundation.....	79
9.2.1	NetworkMessage.....	80
9.2.2	DataSetMetaData	82
9.2.3	DataSetMessage.....	85
9.2.4	DataSetMetaDataType	88
9.2.5	ConfigurationVersionDataType	90
9.2.6	FieldMetaData.....	90
9.2.7	LocalizedText.....	94
9.2.8	KeyValuePair.....	95
9.2.9	StructureDescription.....	95
9.2.10	StructureDefinition.....	95
9.2.11	StructureField	96
9.2.12	EnumDescription.....	98
9.2.13	EnumDefinition	98
9.2.14	EnumField	99
9.2.15	SimpleTypeDescription	99
9.2.16	ServiceNetworkMessage.....	100
9.2.17	ServiceParametersRequest	102
9.2.18	CallMethodRequest	102
9.2.19	ServiceParametersResponse.....	103
9.2.20	CallMethodResult	103
9.3	OPC UA Objects, Defined by the OI4 Alliance	105
9.3.1	MAM (Master Asset Model).....	105
9.3.2	Health	105
9.3.3	Config.....	107
9.3.3.1	Config (Pub).....	108
9.3.3.2	Config (Set)	116
9.3.4	License	124

9.3.5	LicenseText	126
9.3.6	RtLicense.....	127
9.3.7	Data	127
9.3.8	Metadata	128
9.3.9	Event	129
9.3.9.1	Status.....	130
9.3.9.2	Syslog.....	132
9.3.9.3	NAMUR NE107	133
9.3.9.4	Generic.....	137
9.3.10	Profile	138
9.3.11	PublicationList.....	139
9.3.12	SubscriptionList.....	145
9.3.13	Interfaces	146
9.3.14	ReferenceDesignation	146
9.3.15	Pagination.....	152
9.3.15.1	PaginationRequest.....	153
9.3.15.2	Pagination.....	154
9.3.16	Locale.....	156
9.4	OPC UA Methods, Defined by the OI4 Alliance	156
9.4.1	FileUpload	157
9.4.2	FileDownload.....	157
9.4.3	FirmwareUpdate	157
9.4.4	Blink.....	157
9.4.5	NewDataSetWriterId	158
10	Message Bus Communication	160
10.1	Resources	160
10.1.1	MAM (Master Asset Model).....	162
10.1.2	Health	166
10.1.3	Config.....	169
10.1.4	License	179
10.1.5	LicenseText	183
10.1.6	RtLicense.....	186
10.1.7	Data	189
10.1.8	Metadata	196
10.1.9	Event	200
10.1.10	Profile	204
10.1.11	PublicationList.....	207
10.1.12	SubscriptionList.....	215
10.1.13	Interfaces	222

10.1.14 ReferenceDesignation	223
10.2 Common Services	231
10.2.1 FileUpload	231
10.2.2 FileDownload	234
10.2.3 FirmwareUpdate	237
10.2.4 Blink	240
10.2.5 NewDataSetWriterId	242
10.3 Specific Services	245
10.3.1 Read	245
10.3.2 Write	245
10.3.3 Subscribe	246
10.3.4 Unsubscribe	246
10.3.5 GenericMethod	246
11 The Open Edge Computing Platform (MVP)	247
11.1 Minimum Requirements on an Alliance Compliant OEC Platform	248
11.2 Minimum Requirements on an OI4 Alliance Compliant Application	249
11.3 Legal Requirements & Software Licenses	249
Appendix A	251
A 1 The OEC Registry	251
A 2 Predefined DataSetClassIds for Resources of the Alliance	253
A 3 The dnp-Encoding	256
A 4 Glossary	261
A 5 Authors	264
A 6 List of Figures	265
A 7 List of Tables	267
A 8 History	268
Appendix B	279
B 1 Examples for Message Bus Communication	279
B 1.1 Resources	279

B 1.1.1	MAM	279
B 1.1.2	Health.....	287
B 1.1.3	Config.....	291
B 1.1.4	License.....	327
B 1.1.5	LicenseText	336
B 1.1.6	RtLicense	342
B 1.1.7	Data.....	342
B 1.1.8	Metadata	349
B 1.1.9	Event	352
B 1.1.10	Profile.....	356
B 1.1.11	PublicationList	360
B 1.1.12	SubscriptionList.....	372
B 1.1.13	Interface.....	382
B 1.1.14	ReferenceDesignation.....	382
B 1.1.15	Pagination.....	392
B 1.1.16	Locale.....	396
B 1.2	Common Services	398
B 1.2.1	FileUpload	398
B 1.2.2	FileDownload.....	399
B 1.2.3	FirmwareUpdate	399
B 1.2.4	Blink.....	399
B 1.2.5	NewDataSetWriterId.....	399

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.”*

RECOMMENDED → normal recommendation

Example: *“To prevent access by unauthorized communication participants, the operator must prevent this via ACL.”*

HIGHLY RECOMMENDED → strong recommendation

Example: *“Do not store any private keys here!”*

ATTENTION → something to consider

Example: *“Information in this chapter requires alignment with other working groups and has not been maturely specified.”*

The following style with line numbering is used in case code or command line input/output is used:

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>
```

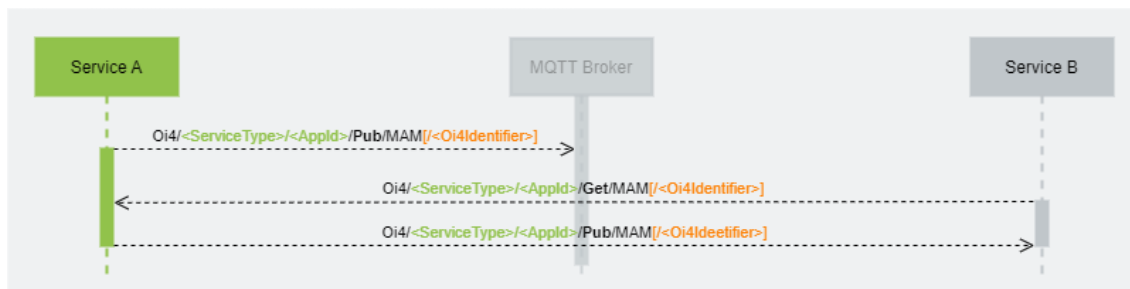
References to elements used in messages or topics of the Message Bus communication are highlighted in gray background.

Example:

MessageType

For the illustration of interactions, UML sequence diagrams are used.

Example:



All JSON definitions, given either by the OPC Foundation or by the OI4 Alliance are strictly written in PascalCase.

NOTE A good source for Capitalization Conventions can be found on [Microsofts pages](#).

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": "1639693510000-ITConnector/consumer.com/MEService/MES%2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 3875,
      "Filter": "",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Health": "NORMAL_0",
        "HealthScore": 100
      }
    }
  ]
}
  
```

2 Introduction to Open Edge Computing - Considered Standards & 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 introduces four layers: Open Edge Connectivity, Open Edge Computing, Open Operator Cloud platform, and Common Cloud Central. This guideline 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.

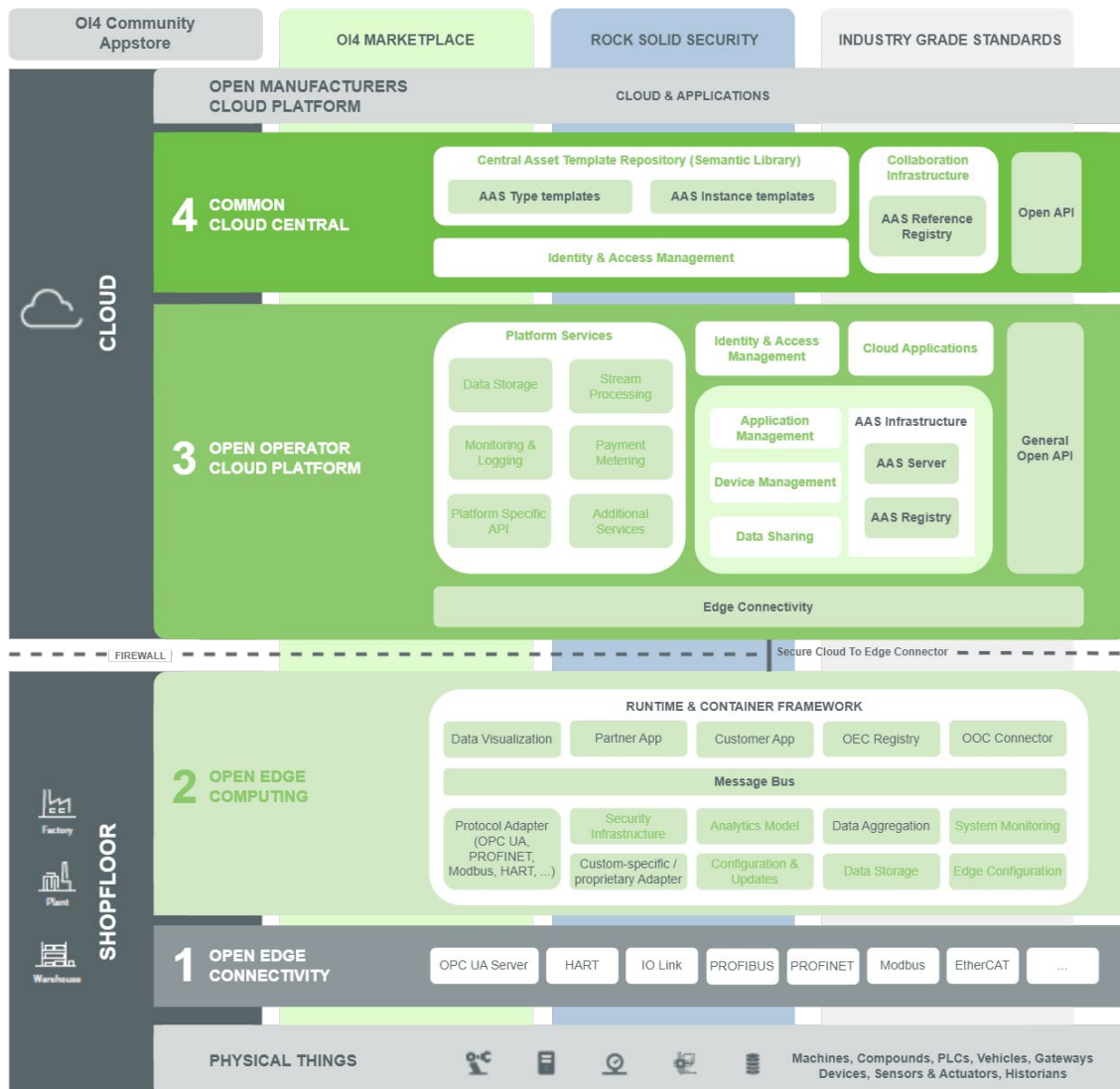


Figure 1: Open Industry 4.0 Alliance Reference Architecture

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. Furthermore, a Message Bus is a mandatory component in the Open Edge Computing layer, ensuring standardized communication and interaction between the various 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. To resolve potential ambiguities in MQTT topic definitions, the Open Industry 4.0 Alliance defines topic structures, ensuring interoperability among 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. A 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 widely regarded as best practice within the industry, particularly concerning device information outlined in [OPC UA Part 100](#). These definitions have also been adopted by other industrial groups.

The mentioned technologies support the Open Industry 4.0 Alliance's solutions developed according to this guideline in meeting interoperability requirements. However, to address and distinguish components effectively, a unified approach is needed that goes beyond the neutral technology definitions of the used standards. For this purpose, an *Di4Identifier* and a *Master Asset Model* are defined as mandatory components for every entity that interacts with the Message Bus. These are core concepts of the Open Industry 4.0 Alliance and are described in the following chapters.

Regarding cyber security, the [IEC 62443](#) standard is considered for the overall automation system. For the edge computer, the [DIN SPEC 27070](#) standard provides specialized considerations.

2.1 Introduction to Edge Computing

Edge Computing is a decentralized data processing paradigm that occurs at the edge of the network, supporting the cloud in business-critical scenarios. In the context of manufacturing, Edge Computing is applied at the shop floor to enable continuous synchronization with the cloud.

The edge computer 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. The cloud and edge infrastructure enables the distribution of applications and data to the optimal location, ensuring seamless business scenario continuity. 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 decision making.
- For sensitive data, Edge Computing provides a way to keep data in the premises.

The cloud typically manages the edge computer, while also operating, managing, and orchestrating applications running at the edge in most cases.

NOTE *It is important to differentiate Edge Computing or edge computers to traditional on-premise systems. While Edge Computing requires a connection to cloud systems, traditional on-premise systems or solutions do not necessarily need to be connected to the cloud.*

From a cyber security 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). The [MQTT v3.1.1](#) standard is stable and particularly provides mechanisms for a secured communication. [MQTT](#) is well supported through various libraries and

implemented in several programming languages. Both, open source and commercial offerings, based on the open MQTT 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 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.*

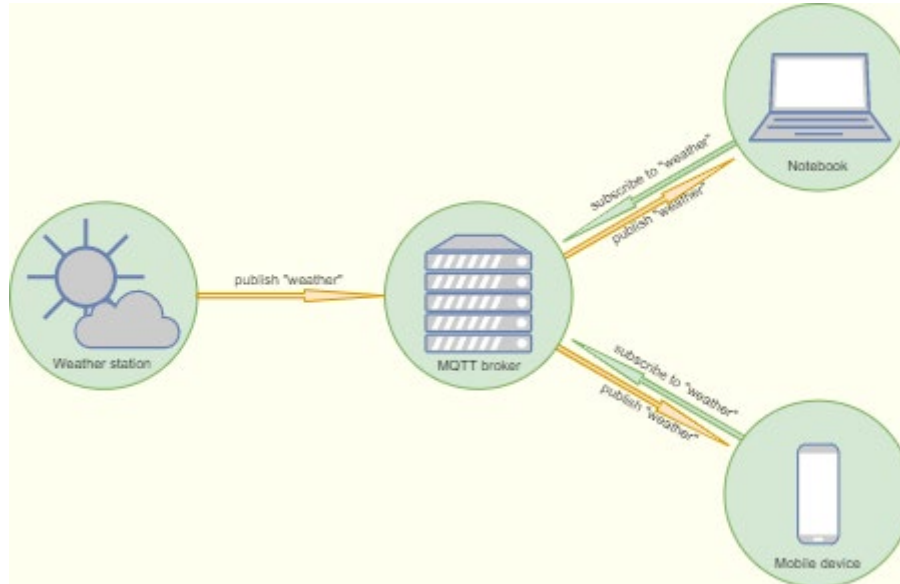


Figure 2: How the Pub/Sub mechanism functions

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: opconnect.opcfoundation.org

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 (ref. <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 time-sensitive networking (TSN).

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 crashing of a container is isolated from other containers at the same host consequently avoiding 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.

The 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 container 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 about container technology can be found here:

<https://www.docker.com/resources/what-container>.

Open Industry 4.0 Alliance specific requirements are available in chapter [6](#) of this document.

2.5 Cyber Security

The [IEC 62443](#) is a standard addressing [cyber security](#) topics of the automation system as a 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 series of standards which focus on cyber security of industrial automation and control systems. They cover 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
0	No special requirement or protection required
1	Protection against casual or coincidental violation

Security level	Description
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

NOTE The means described in this guideline aim to fulfill the 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.

Within the Open Industry 4.0 Alliance, a security gateway refers to an edge computer which connects OT and IT.

The security relevant requirements in [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 of the four security levels of [IEC 62443](#).

Security gateway profile	Description
Basic	The "Basic" profile is primarily intended for scenarios where there is a limited need for security, such as for demonstration or testing purposes. The "Basic" profile is generally mapped to security level 1 of IEC 62443.

Security gateway profile	Description
Trusted	<p>The "Trusted" profile is primarily intended for scenarios where the protection of the processed and transmitted data is essential.</p> <p>The "Trusted" profile is generally mapped to security level 3 of IEC 62443.</p>
Trusted Plus	<p>The "Trusted Plus" profile provides an additional layer of protection against manipulation by administrators who may attempt using technical means within the "Trusted" profile.</p>

Table 2: DIN SPEC 27070 - Security gateway profile

NOTE *The means described in this guideline aim to fulfill the security profile labeled as "Basic".*

3 The Oi4Identifier

The `Oi4Identifier` is a basic element of any Open Industry 4.0 Alliance system. It is mandatory for every asset and software to include this identifier, which enables the unique identification of assets across different systems. Unless the manufacturer defines the `ProductInstanceUri` the same as the `Oi4Identifier`, it can be generated at any time of the assets life cycle and serves as a stable point of reference for that particular asset in all interactions 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 life cycle.

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 it may differ from other existing or upcoming vendor-specific IDs, often referred to as "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 (Uniform Resource Identifier) and can be divided into its individual 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: String as given by the manufacturer but must necessarily be written in lowercase letters.

Access: Read only

Requirement: Mandatory

The `ManufacturerUri` is the part of the manufacturer URL were the scheme (e. g. `<https://>`) and the third level domain (e. g. `<www.>`) are excluded. 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 `ManufacturerUri` must necessarily be written in lower case. Upper cases are not allowed.*

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 is considered necessary. The only requirement is that the site is under control of the manufacturer and the provision of data is not required.*

`Model`:

Type: dnp-encoded ([A3](#)) string
Access: Read only
Requirement: Mandatory

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: dnp-encoded ([A3](#)) string
Access: Read only
Requirement: Mandatory

Contains a product code, often called order number, which allows the manufacturer to explicitly 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: dnp-encoded ([A3](#)) string
Access: Read only
Requirement: Mandatory

Contains 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 *When deploying a container which by its nature has no serial number and can be instantiated multiple times, use the container name as `SerialNumber` as defined in [6.2](#).*

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 the 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 three forward slashes and the dnp-encoded components `ManufacturerUri`, `Model`, `ProductCode` and `SerialNumber` must not exceed 255 characters.*

NOTE *Requirement 5.4 (forbidden 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.*

An `Oi4Identifier` looks like this at the end:

```
ManufacturerUri/dnp-encoded(Model)/dnp-encoded(ProductCode)/dnp-  
encoded(SerialNumber)
```

3.2 Additional Provisions Regarding the Oi4Identifier

The `Oi4Identifier` as a lasting reference to a particular asset must fulfill many roles in an Open Industry 4.0 Alliance system. For this purpose, 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](#). It is expected to meet all the requirements outlined in that specification, as well as the requirements specified 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 `SerialNumber` is then sufficient to identify and distinguish the product.

To enable local generation of the `Oi4Identifier` in the Open Edge Computing layer, 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 provide 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.

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 these IDs with the `Oi4Identifier` because different IDs might be used in different contexts.

4 The Master Asset Model

To provide further standardized 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 especially useful 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

This describes 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's name regarding all legal appendices and so forth. In order to avoid shortcomings in identification, 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 OI4 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 Model belongs to. Note that model names often are not sufficient to unambiguously identify a product. In order to avoid shortcomings in identifications, 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 life cycle 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 OI4 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 OI4 Alliance does not define the content of this field, since it belongs to the provider of the asset. However, best practice for OI4 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

This property 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 of 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 numbers 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 the Open Edge Computing Level.

NOTE Each component, regardless if it is a physical thing or an application, which was detected by the OI4 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 chapter 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 OI4 Reference 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 an 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 on the implementation of the Asset Administration Shell (ASS), the onboarding sequence may look like shown in [figure 3](#). This sequence diagram gives an overview of the default flow. There may be additional/alternative mechanisms and applications. The blocks of the sequence diagram are following the naming of the OI4 Reference Architecture as shown in [figure 1](#).

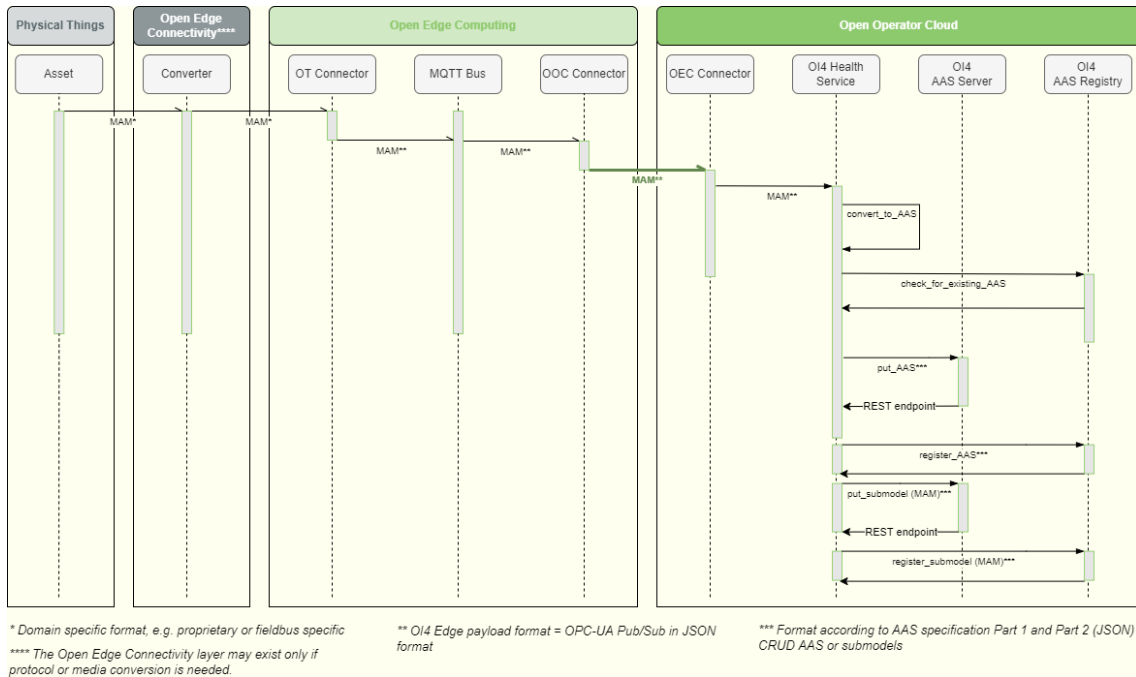


Figure 3: Sequence diagram for asset onboarding

NOTE The Open Industry 4.0 Alliance does not affect the communication of physical things or the conversion from proprietary physics/protocols such as RS485/Modbus RTU to ethernet-based communication required by the Open Edge Computing layer. Therefore, the physical things and the conversion on the Open Edge Connectivity layer are domain specific and out of scope for the OI4 Alliance.

NOTE An example of a media converter would be the integration of HART or wireless HART devices into an OEC environment via a HART-to-HART-IP gateway.

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 OI4 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. The blocks of the sequence diagram are following the naming of the OI4 Reference Architecture as shown in [figure 1](#).

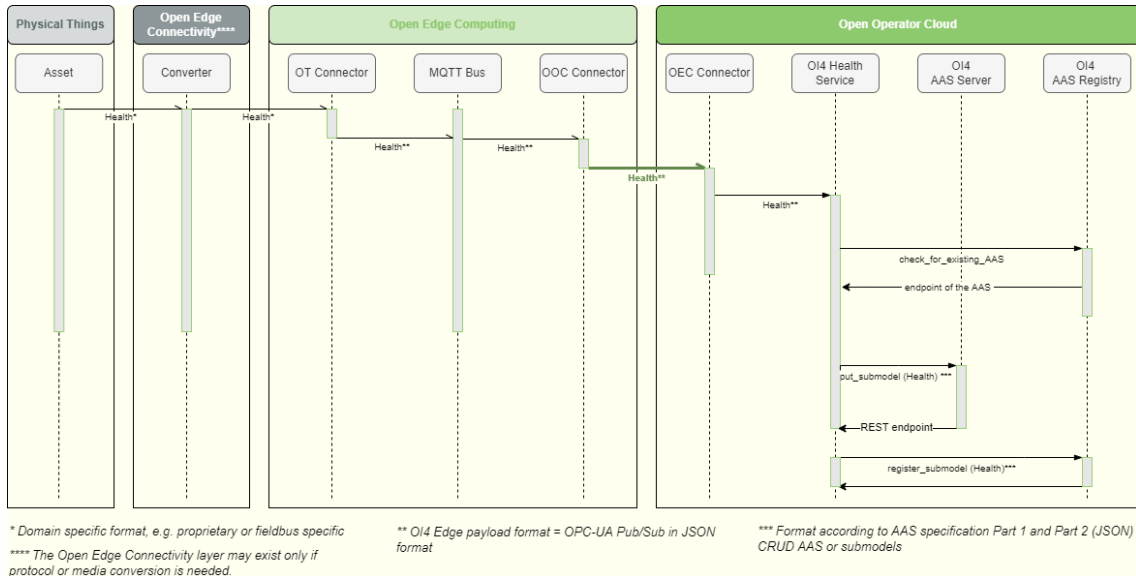


Figure 4: Sequence diagram for condition monitoring (Health)

NOTE The Open Industry 4.0 Alliance does not affect the communication of physical things or the conversion from proprietary physics/protocols such as RS485/Modbus RTU to Ethernet-based communication required by the Open Edge Computing layer. Therefore the physical things and the conversion on the Open Edge Connectivity layer are domain specific and out of scope for the Alliance.

NOTE An example of a media converter would be the integration of HART or wireless HART devices into an OEC environment via a HART-to-HART-IP gateway.

5.1.3 Handling Process Data

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

The handling of process data in the OI4 Reference Architecture can be done in very different ways. Data can be stored, aggregated or connected in a time series, but it is not mandatory to do so. The technical committee of the OI4 Alliance has devised sequence diagrams for three scenarios which are likely to 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.

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 format.

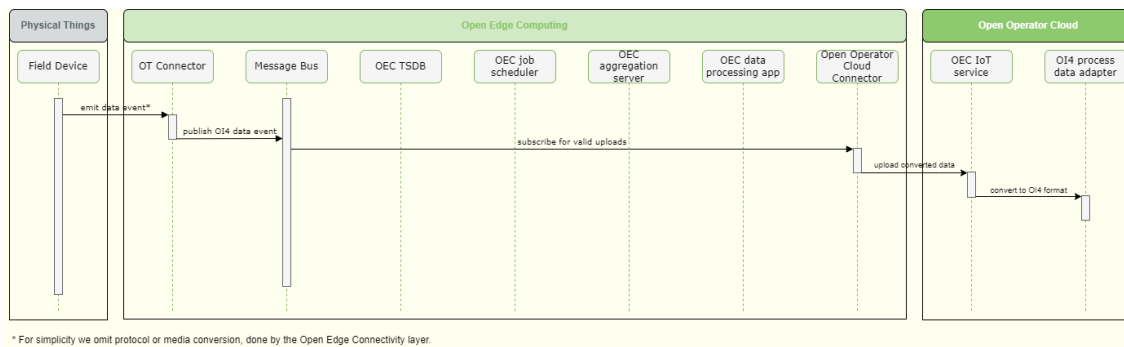


Figure 5: Plain process data sequence diagram

The second use case repeats the steps from the first one but adds the task of edge data processing. In the Open Edge Computing environment, data processing applications can receive the incoming data, process it based on their configuration, and then publish the resulting data for uploading to the Open Operator Cloud platform through the Message Bus.

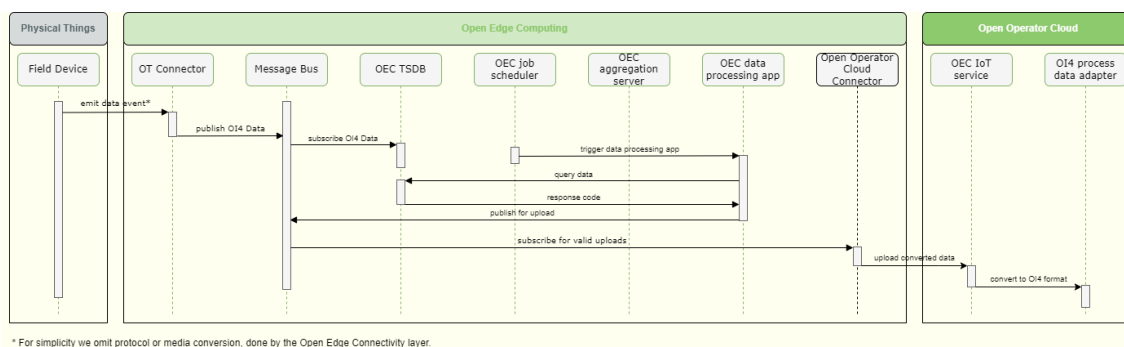


Figure 6: Edge data processing sequence diagram

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.

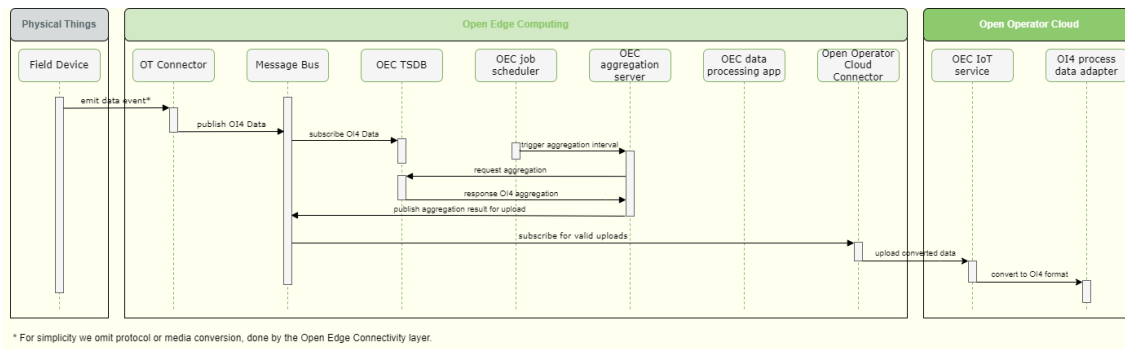


Figure 7: Store and forward sequence diagram

5.2 Detailed Container Interactions Inside the Open Edge Computing Layer

The Open Edge Computing layer consists of containers that perform various tasks related to the use cases of the OI4 Alliance. 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 three use cases “onboarding of a field device” (5.2.1), “condition monitoring” (5.2.2) and “process data ingestion” (5.2.3).

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 specific OT connector 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, manufacturer URI, etc.

Once the Master Asset Model is created, the OT connector 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.

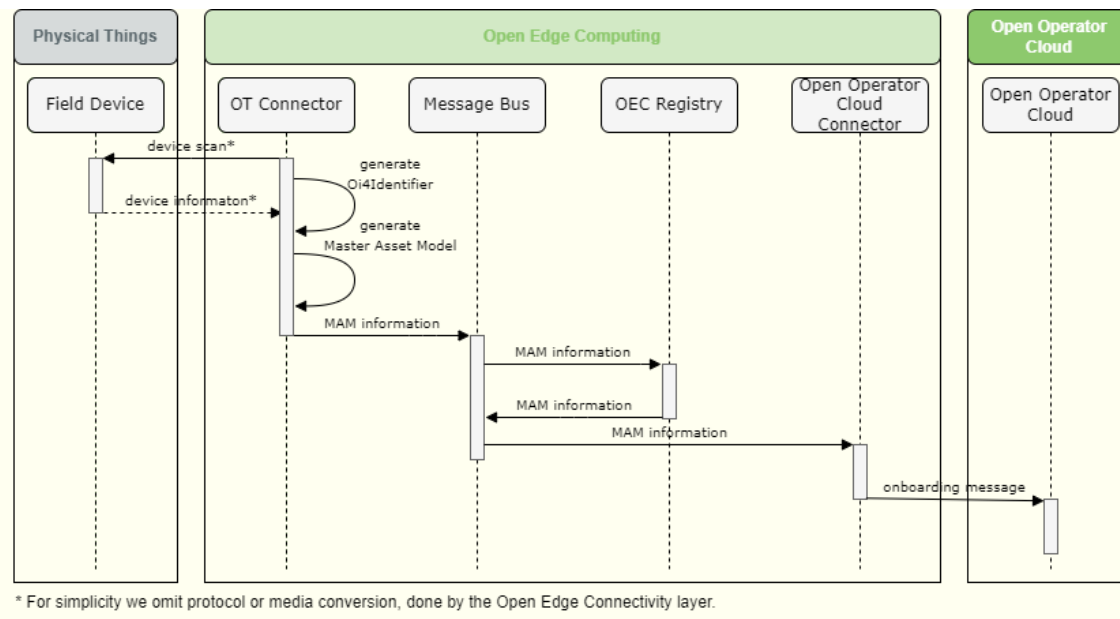


Figure 8: Sequence diagram for the onboarding of a field device

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 OT connector sends a `Health` message (10.1.2) to the Message Bus. This message can be used by all containers listening to `Health` messages to monitor 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.

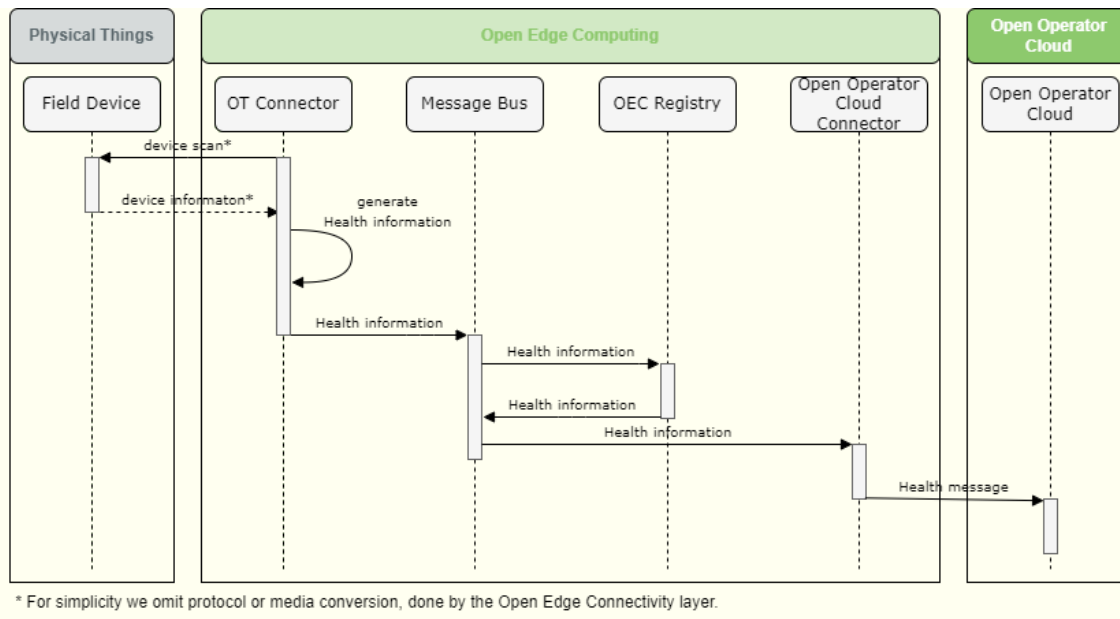


Figure 9: Sequence diagram for the condition monitoring (Health) of a field device

5.2.3 Process Data Ingestion of a Field Device

ATTENTION Information in this chapter requires alignment with other work 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 OT connector publishes the ingested process data as a `Data` object (10.1.7) on the Message Bus. The `Data` contains all source information for the process data within its topic.

The Open Operator Cloud connector receives the new `Data` 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.

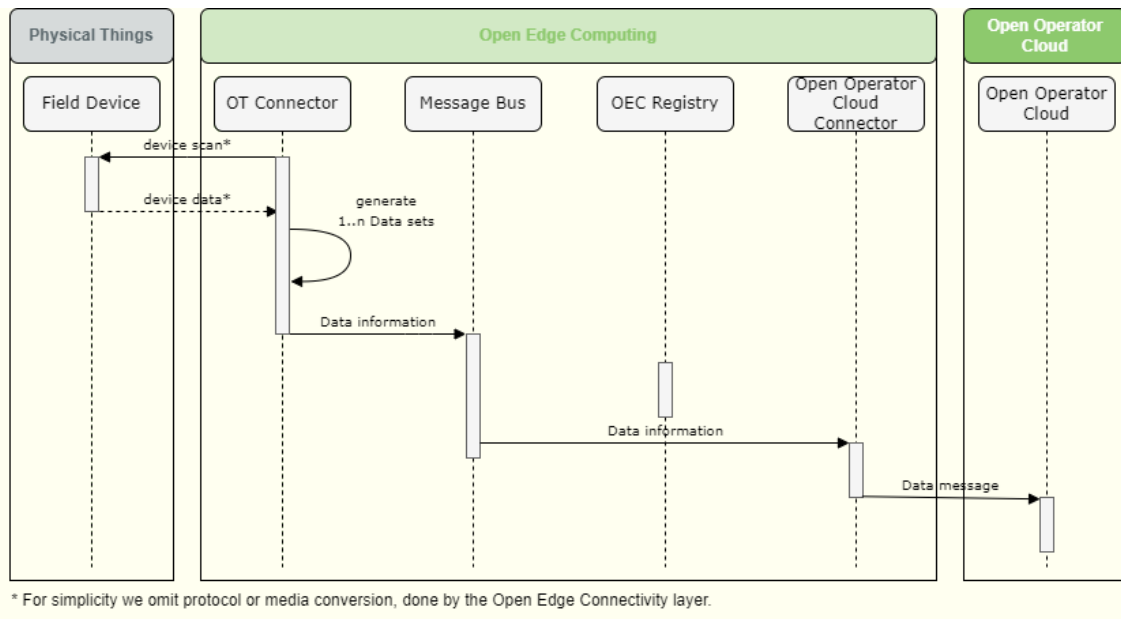


Figure 10: Sequence diagram for the process data ingestion of a field device

6 Container Environment

To enable all Open Industry 4.0 Alliance members to interoperate on Open Edge Computing platform with each other, the OI4 Alliance has to define the expectation from the container's 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 that 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 side of the host. It just specifies how it is mapped to the container to clearly describe 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 HIGHLY RECOMMENDED <i>Do not store any private keys here!</i>

Table 3: Message Bus storage path

NOTE The target path is the path inside the container, which needs to be mapped to a host-defined path inside the host system.

RECOMMENDED The host path shall be mounted read only!

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 <i>If no local CA is present, the global CA (see 6.1.2) is used.</i>
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 address or hostname>

Type: String

Requirement: Mandatory

Defines the IP address or host name of the MQTT broker used.

SecurePort:

Value range: <UINT16>

Type: UINT16

Requirement: Mandatory

Defines the port, where the MQTT broker is listening.

MaxPacketSize:

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.

The minimum value of MaxPacketSize is 262,144 bytes (256 kiB), the theoretical maximum is INT32_{max}.

NOTE *The intention of the MaxPacketSize is to protect the Message Bus broker and client.*

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. HIGHLY RECOMMENDED <i>Do not store any private keys here!</i>

Table 5: OI4 Alliance CA storage path

NOTE *The target path is the path inside the Docker container, which needs to be mapped to a host-defined path inside the host system.*

RECOMMENDED *The host path shall be mounted read only!*

The certificate 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 Alliance 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 perspective of the container.

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 a base64 encoded username and password, which is used for MQTT authentication in the following format: <code>base64 (<username>:<password>)</code>
/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

NOTE The target path is the path inside the Docker container, which needs to be mapped to a host-defined path inside the host system.

RECOMMENDED *The host path shall be mounted read only!*

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
```

NOTE *Secrets mounted directly in the `secrets` folder cause issues in a Kubernetes environment as Kubernetes also mounts the `run/secrets` folder. A known solution for this problem is the use of [subPaths](#). An example of the usage is shown below:*

```

...
kind: deployment
...
spec:
  template:
    ...
    spec:
      containers:
        ...
        volumeMounts:
          - name: secrets
            mountPath: "/run/secrets/mqtt_credentials"
            readOnly: true
            subPath: mqtt_credentials
      volumes:
        - name: secrets
          secret:
            secretName: mqtt-credentials
...

```

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 paths 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 Alliance 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.

NOTE *The target path is the path inside the Docker container, which needs to be*

mapped to a host-defined path inside the host system.

RECOMMENDED *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 access to read/write 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 OI4 Alliance-compliant container shall have a well-defined name for identification. 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 OI4 Alliance-compliant containers is to have a unique `oi4Identifier` (3), which makes it necessary to have a serial number or an equivalent for that.

To combine these needs, the container name must be unique worldwide according to the definition of the OI4 Alliance but is also not allowed to exceed 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 a 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. the OEC Registry container might be named the same in each system). With that in mind, all instances of a specific application end up 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 to assign a host name must also be followed for container names so that the DNS resolution is working.*

An example of how the container name and host name can be set in the 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 which 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 computers, 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 OI4 Alliance is permitted and the default.

RECOMMENDED *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 connected to 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 OI4 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.

Advantages	Disadvantages
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: Advantages/disadvantages - 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 a key-value store must be 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 networks, the host sends all communication messages over named pipes. However, this method can lead to an increased security risk, as all traffic passes through the same containers without any separation mechanisms.

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

NOTE *For the use of containers in an OI4 Alliance-compliant Edge Computing environment, the use of bridge network types should be preferred.*

7 General Requirements on MQTT

The Open Edge Computing framework is using MQTT as Message Bus. This Message Bus is used as the standard communication channel between OI4 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 OI4 Alliance.

The following describes the framework conditions and default behavior for an MQTT client that is to be used in the OI4 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 OI4 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 work groups and the technical committee of the Open Industry 4.0 Alliance for further technology steps.

The chosen MQTT standard provides members of the OI4 Alliance with the opportunity to use following key features:

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

Additionally, the OI4 Alliance is taking care about:

- OT related topic namespace definitions, which are service oriented
- OT related payload definitions, using OPC UA PubSub 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 OI4 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 host name 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. Through the Message Bus storage, the default might be set to a system-dependent value (6.1.1). Non-secure connections through port 1883 is not supported.

All data transmissions between assets and the Open Edge Computing layer (and possibly some going beyond that, 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 version 1.2 or higher is mandatory). The broker preferably shall use a certificate which is signed by a well-known or user specific certificate authority (CA).

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 itself to the Message Bus either by username and password or with a 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 private key for an existing client certificate and username/password is available, the client certificate based authentication is used.

NOTE *The username must only contain unreserved characters according to the [RFC3986](#). These include uppercase and lowercase letters, decimal digits, hyphen, period, underscore, and tilde (a-z, A-Z, 0-9, -, ., _, ~). In addition it is also allowed to use the @ symbol.*

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 (ClientId) for the MQTT client. To avoid problems with length restrictions in some rare MQTT implementations, a name of the container should be used which is unique, present and accessible in any OI4 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

The Clean Session flag (boolean) represents a feature for the subscriber which 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 the 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-ranged 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 behaviors 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.

When it comes to Open Edge Computing, where there is always a strong connection on local networks, using QoS 0 as the default level should be sufficient.

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 OI4 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` ([8.1.5.1](#)) 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 follows 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 Alliance 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 the topic resource `Health` ([8.1.5.1](#)) 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` ([8.1.5.1](#)) 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` ([8.1.5.1](#)).*

8 Topics Definition

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

A whole topic is a UTF-8 encoded 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 it 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 allowed.

8.1 Topic Namespace Elements

For better interoperability of independently working applications from different members of the OI4 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 Edge Computing Development Guideline, use the following schema:

```
Oi4/<ServiceType>/<AppId>/<Method>/<Resource>[[/<Source>]/<Filter>]
```

NOTE As described in the following sections, the elements `Source` 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-compliant 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 OI4 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.

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 `ServiceTypes` 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 associated with it. An `OTConnector` likely makes assets available through the Message Bus, which can be listed from `Registry` and onboarded from `OOConnector`. 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 services.

Persistence:

Provides mechanisms to persist data. It is not defined how the persistence will work; the `ServiceType` just clarifies 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 appropriate way. The sources function similar to other applications by providing raw data, which is then aggregated to generate enriched data.

OCCConnector:

Communicates with the Message Bus on one side and with a 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 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` and others.

`Get`:

Is used for the dedicated request of a `<Resource>` such as `Data`, `MAM`, `Health` and others. 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` with the `<EventCategory>` `Status` and the `<EventLevel>` `Good`, `Uncertain` or `Bad`.

Del:

Used to delete `ReferenceDesignation` or entry of `SubscriptionList`.
A `Del` should cause an `Event` with the `<EventCategory>` `Status` and the `<EventLevel>` `Good`, `Uncertain` or `Bad`.

All Methods above are meant for `Pub/Get/Set/Del` standard Resources ([8.1.5.1](#)) which are provided by applications or devices. Furthermore, the 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 committee discussed other technical solutions just as REST but defined that the actual function set should be available over a single technology stack which is 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, the 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 committee discussed other technical solutions just as REST, but defined, that the actual function set should be available over a single technology stack which is 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](#)). 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 also be available via System Logging Protocol (syslog) and might be used for audit-trailing. In this case, the category `syslog` ([9.3.9.2](#)) is used. Other event categories are available as listed in [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 and when the firmware update is performed is domain specific and handled by the application providing 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

Specific services are domain-dependent. That means each of these 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 publicly 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 Source Element

The `Source` always describes the asset providing the information. Therefore the `Source` is the `Oi4Identifier` of the related application or device.

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.*

NOTE *The Filter shall not include characters, which are used as special characters in MQTT topics. Therefore, a Filter is an UTF-8 string, without the characters +, # and /. The OI4 Alliance describes this encoding <Topic encoded String>. If those characters shall be used, an URL encoded string might be best practice.*

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: `<Topic 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 OI4 Alliance defined a DataSet called `Oi4Pv` (see [9.3.7](#)), which can be used in addition to provide standardized access to the process values.

Resource: Config

For the resources `Config`, the `Filter` contains the name of the desired data set.

`<Context.Name>`:

Value Range: `<Topic encoded String>`

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 DataSet.

`<LicenseId>`:

Value Range: `<Topic encoded String>`

Type: String

The `LicenseId` filter can be used to filter messages for a particular DataSet.

NOTE *As a best practice, the OI4 Alliance recommends naming the `LicenseId` according to [SPDX \(ISO/IEC 5962:2021\)](#).*

Resource: Event

For the resource `Event`, the `Filter` is used to describe the event category and the event level and is defined as following:

<EventCategory/><EventLevel>:

Value Range: <EventCategory followed by separator and EventLevel>

Type: String

The <EventCategory> describes the category such as Syslog, Status, Net107 or Generic. The EventLevel describes the level inside the specified category, e.g. Debug or Warning for Syslog or Good or Bad for Status.

Resource: Event, EventCategory: Syslog, EventLevel: LogLevel

<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 a 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 requires 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 requires 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 took 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.*

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, EventCategory: Status, EventLevel: StatusCode

<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, EventCategory: Ne107, EventLevel: StatusSignal

<StatusSignal>:

Value Range: <NAMUR NE107 status signal>

Type: String

The `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, EventCategory: Generic, EventLevel: GenericLevel

<GenericLevel>:

Value Range: <GenericLevel>

Type: String

The `GenericLevel` 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)
- `Source` (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)
- ReferenceDesignation (9.3.14)

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 on 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)
- `Source` (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 using OPC UA-conform JSON format for payload definition.

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

OPC UA PubSub is described in [Part 14](#) (PubSub).

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

For a better interoperability of existing and future OI4 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>/<Source>/<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>",
      "Source": "<Oi4Identifier>",
      "Payload": {
        <add your DataSet here>
      }
    }
  ]
}
```

Several examples of a 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:

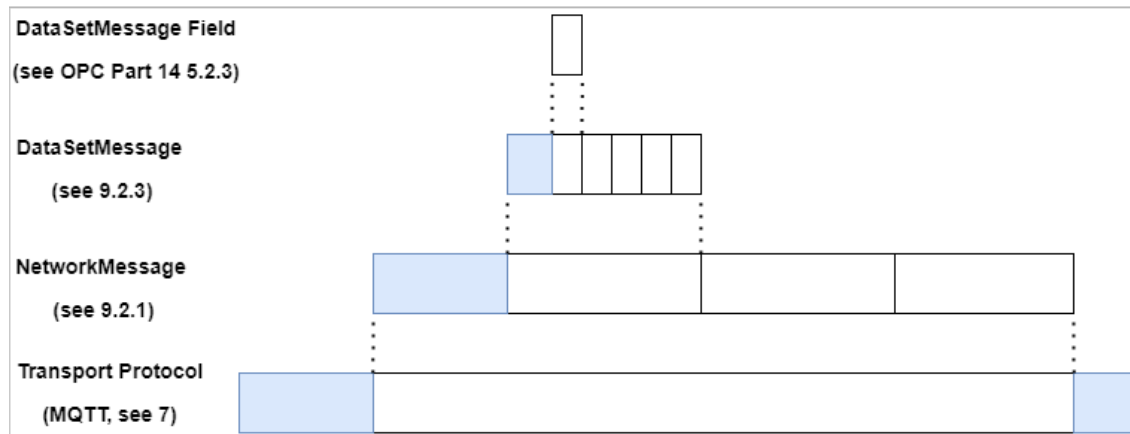


Figure 11: OPC UA PubSub message embedded in the transport protocol

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](#), the Open Industry 4.0 Alliance defined the additional `<MessageType> MSG`. This type offers the possibility to implement a call/reply 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 OI4 Alliance-defined or user-defined payload.

[Figure 12](#) illustrates a non-exhaustive example of the structure of a `ua-data` message encoded in JSON:

```

{
  "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": <UINT32>,
      "Filter": "<Filter>",
      "Source": "<Oi4Identifier>",
      "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](#).

[Figure 13](#) illustrates a 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>",
  "Source": "<Oi4Identifier>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "MetaData":
  {
    "Name": "<String>", DataSetMetaData (see 9.2.4)
    "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 OI4 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 PubSub specification, because OPC UA does not define or allow other `<MessageTypes>` types besides `ua-data` and `ua-metadata` for PubSub. OPC UA client/server does define a `<MessageType>` `MSG`, but it is not used for PubSub and contains many keys, which are not relevant for the PubSub 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:

```

{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<ServiceType>/<AppId>",
  "DataSetClassId": "<GUID>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Message": {
    "MethodsToCall": [
      {
        "MethodId": "<MethodName>",
        "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:

```

{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "MSG",
  "PublisherId": "<ServiceType>/<AppId>",
  "DataSetClassId": "<GUID>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Message": {
    "Results": [
      {
        "StatusCode": "<StatusCode>",
        "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:

`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 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 (e.g. `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](#) and [8.1.3](#)) - separated by a `"/`.

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

`DataSetClassId`:

Value Range: `<GUID>`

Type: GUID which has TypeOf String

Requirement: Optional

The definition of the GUID, according to the OPC UA [Part 6-5.1.3](#), is defined as a 16 Byte JSON string with separators ([Part 6-5.4.2.7](#)). 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 OI4 Alliance or other standards and refers to related metadata.

For some recurring use cases, such as `MAM`, fixed GUIDs are specified from the OI4 Alliance and must be used ([A2](#)).

NOTE The `DataSetClassId` shall be present for all resources defined by the Open Industry 4.0 Alliance which have `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](#).

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:

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 (e.g. `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](#) and [8.1.3](#)) - 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`. 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](#). In combination with the used resource in the topic, the `Filter`, together with the `Source`, 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](#).

NOTE The `Filter` shall be a topic encoded string as explained in [8.1.7](#).

Source:**Value Range:** <Oi4Identifier>**Type:** String**Requirement:** Mandatory

The **Source** is mandatory, but does not belong to OPC UA **DataSetMessage** according to [Part 14-7.2.3.3](#). In combination with the used resource in the topic, the **Source**, together with the **Filter**, contains the readable reference to the **DataSetWriterId** and is identical to the **Source** in the topic (8.1.6) if present. The **Source** 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](#).

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](#)):

`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`.

A range of `DataSetWriterIds` are reserved for special use cases. A `PaginationRequest` ([9.3.15.1](#)) is always using `DataSetWriterId` 1, a `Pagination` ([9.3.15.2](#)) uses the 2 and a `Locale` ([9.3.16](#)) uses the 3.

The application starts with `DataSetWriterIds` from 10, because the `DataSetWriterIds` up to 9 are reserved for special use cases.

`SequenceNumber`:

Value Range: `<UINT32>`

Type: `UInt32`

Requirement: Optional

A number which is strictly increasing in sequence and 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 or 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

Status code to be used as defined in OPC UA [Part 4-7.34.2](#) and [CSV-File](#).

NOTE *The `Status` is not required and should not be sent when the status is OK. If the `Status` is not equal to OK, we use the status codes provided by the OPC Foundation.*

Filter:

Value Range: `<Filter>`

Type: String

Requirement: Conditional

Depending on related use cases, the `Filter` might be mandatory or optional, but does not belong to OPC UA `DataSetMessage` according to [Part 14-7.2.3.3](#). In

combination with the used resource in the topic, the `Filter`, together with the `Source`, 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 `DataSet` in the `Payload` with the related source. In OPC UA context, this is done via `DataSetWriterId`. Though, it is not very intuitive and might need additional actions to get missing information via `PublicationList`, defined in 9.3.11.

NOTE The `Filter` shall be a topic encoded string as explained in 8.1.7.

NOTE Several resources such as `MAM` or `Health` and others do not make use of `Filter` in a `Message Bus` topic and `DataSetMessage`.

`Source`:

Value Range: `<Oi4Identifier>`

Type: String

Requirement: Mandatory

The `Source` is mandatory, but does not belong to OPC UA `DataSetMessage` according to Part 14-7.2.3.3. In combination with the used resource in the topic, the `Source`, together with the `Filter`, contains the readable reference to the `DataSetWriterId` and is identical to the `Source` in the topic (8.1.6), if present. The `Source` always describes the asset providing the information. Therefore, the `Source` is the `Oi4Identifier` of the application or device.

NOTE The `Source` 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: `<DataSet>`

Type: `DataSet` 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 14-6.2.3.2.2](#)):

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>`

Type: GUID which has `TypeOf String`

Requirement: Optional

The definition of the GUID, according to the OPC UA [Part 6-5.1.3](#), is defined as a 16 Byte JSON string with separators ([Part 6-5.4.2.7](#)). 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 OI4 Alliance or some other standards.

For some recurring use cases, such as MAM, fixed GUIDs are specified from the OI4 Alliance and must be used (A2).

NOTE The DataSetClassId must be present for all resources defined by the Open Industry 4.0 Alliance, which have 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: EnumDescription 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 14-6.2.3.2.5](#)):

`MajorVersion`:

Value Range: <UINT32>

Type: VersionTime which has TypeOf UInt32

Requirement: Mandatory

Timestamp of metadata definition in seconds defined since January 1st, 2000 ([Part 4-7.38](#)).

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 January 1st, 2000 ([Part 4-7.38](#)).

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 14-6.2.3.2.3](#)):

`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. The default value shall be a null or empty LocalizedText.

FieldFlags:

Value Range: <DataSetFieldFlags>

Type: Subtype of UInt16

Requirement: Mandatory

Flags for the field (see definition in OPC UA [Part 14-6.2.3.2.4](#)).

BuiltInType:

Value Range: <Byte>

Type: Byte

Requirement: Mandatory

`BuiltInType` values are defined in OPC UA [Part 6-5.1.2](#).

NOTE The JSON representation of each `BuiltInType` is defined in OPC UA [Part 6-5.4.2](#).

DataType:

Value Range: <NodeId>

Type: NodeId object

Requirement: Mandatory

JSON representation of `NodeId` is defined in OPC UA [Part 6-5.4.2.10](#).

NOTE *First pitfall is to wonder about missing `Namespace` in a `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 (see OPC UA [Part 14-6.2.3.2.3-Table 7](#) for details).

`ArrayDimensions []`:

Value Range: `<array of UINT32>`

Type: UInt32

Requirement: Mandatory

This field specifies the maximum length of each dimension (see OPC UA [Part 14-6.2.3.2.3-Table 7](#) for details).

`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 which is TypeOf String

Requirement: Mandatory

The unique ID for the field in the `DataSet`. The definition of the GUID, according to the OPC UA [Part 6-5.1.3](#), is defined as a 16 Byte JSON string with separators ([Part 6-5.4.2.7](#)).

Properties []:

Value Range: <array of KeyValuePair>

Type: KeyValuePair (see [9.2.8](#))

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: `LocalId` which is `TypeOf String`

Requirement: Mandatory

The `LocaleId` is used to explicitly identify the language and country/region. It is represented by two lowercase letters for language and two to three uppercase letters for country, separated by a hyphen (e.g. "en-US" or "en-EN"; see OPC UA [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 using 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

The object `QualifiedName` is defined in [Part 3-8.3-Table 24](#) and contains a `NamespaceIndex` and a `Name`.

Value:

Value Range: `<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:** MandatoryJSON 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](#) is a sub type of [DataTypeDescription](#). The object `SimpleTypeDescription` contains the following elements:

BaseDataType:

Value Range: `<NodeId>`

Type: `NodeId`

Requirement: Mandatory

BuiltInType:

Value Range: `<Byte>`

Type: `Byte`

Requirement: Mandatory

DataTypeId:

Value Range: `<NodeId>`

Type: `NodeId`

Requirement: Mandatory

Name:

Value Range: <QualifiedName>

Type: QualifiedName

Requirement: Mandatory

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 defines this 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 (e.g. `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>`

Type: GUID which is TypeOf String

Requirement: Optional

The definition of the GUID, according to the OPC UA [Part 6-5.1.3](#), is defined as a 16 Byte JSON string with separators ([Part 6-5.4.2.7](#)). 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 OI4 Alliance or some other standards.

For some recurring use cases, such as `NewDataSetWriterId`, fixed GUIDs are specified from the OI4 Alliance and must be used ([A2](#)).

NOTE *The `DataSetClassId` shall be present for all resources defined by the Open Industry 4.0 Alliance, that possess 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 the OPC UA call service method (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

The list provides the values of the input arguments. If the list is empty, it means

there are no input arguments. The size and order of this list correspond to the size and order of the input arguments specified by the `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 [Part 4-5.11.2.2-Table 65](#)) 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 OI4 Alliance

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

All the objects are JSON encoded and OPC UA PubSub 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 chapter [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 12](#)).

`HealthScore`:

Value Range: `<Byte>`


Type: Byte in a range of 0..100 %

Access: Read only

Requirement: Optional

`HealthScore` reflects a meter to indicate the 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 about 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	





NE107 status	Definition	Color	Symbol
FAILURE_1	Failure (high severity) Signal invalid due to malfunction in the device, sensor or actuator.	red	
CHECK_FUNCTION_2	Check function (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 12: NAMUR NE107 - definition of symbols

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 OI4 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 “*ETHO 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 “*ETHO settings*” in English or “*ETHO 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 exist 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 exist, 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 OI4 Alliance is missing 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 exist, 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 manner. 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. For example, the Group objects with the name “eth0”, “eth1” and “eth2” are all related to “Network settings”, which is the Context. The Context is related to the Filter of the DataSetMessage, which would be “Network%20settings” in this example.

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.

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 “*ETHO 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 “*ETHO settings*” in English or “*ETHO 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 exist 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 exist, 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 OI4 Alliance is missing 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 exist, 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 manner. 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. For example, the `Group` objects with the name “*eth0*”, “*eth1*” and “*eth2*” are all related to “Network settings”, which is the `Context`. The `Context` is related to the Filter of the `DataSetMessage`, which would be “Network%20settings” in this example.

`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.

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 within 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 parent object.

`LicAuthors []`:

Value Range: `<array of name of author>`

Type: String

Access: Read only

Requirement: Optional (not present, if no author information is 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.

NOTE *The described `DataSetMessage` for `License` contains license information, associated to a specific license. This `License` has a human readable name, called `LicenseId` and is mentioned in `DataSetMessage` as `<Filter>`. Optionally it can be part of the topic as `<Filter>`.*

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 within 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 work 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 OI4 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 OI4 Alliance. Each data object might look different, but it must follow the rules that the OPC Foundation has defined for JSON encoded PubSub `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 OI4 Alliance defines an optional `DataSet` to access the available process values in a standard way. This `DataSetMessage` is called `Oi4Pv` and its key-value-pairs follow a fixed key structure. Key elements are defined below.

NOTE *It is published via `.../Pub/Data/<Oi4Identifier>/Oi4Pv`.*

NOTE *The `Oi4Pv 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 `Oi4Pv` 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. For example, this 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. For example, this 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 `Sv`, 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 OI4 Alliance' resource `Data`, which is 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 OI4 Alliance provides a `DataSetClassId` for this `DataSet` ([A2](#)).

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

`Number`:

Value Range: `<UINT32>`
Type: `UInt32`
Access: Read only
Requirement: Mandatory

A 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 details as syslog entry.

CAT_STATUS_1: Interpret number, description and details as OI4 status

CAT_NE107_2: Interpret number, description and details 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

The **Status** events correspond to the following definition.

The element **Category** for status events is **CAT_STATUS_1**.

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

Number:

Value Range: <UINT32>

Type: UInt32

Access: Read only

Requirement: Mandatory

A number which contains the [OPC UA 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: <SymbolicId>

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 OI4 Alliance defines the `Syslog` event mapping as the following:

`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.

NOTE The optional key Description is not used in context of this Category.

NOTE 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 OI4 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 Part100 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	Check Function (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	Blue	

			a function will soon be restricted due to operational conditions.		
--	--	--	---	--	--

Table 13: 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:

`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 13](#).

`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** event provides information which are undefined and 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:

Number:

Value Range: <UINT32>

Type: UInt32

Access: Read only

Requirement: Mandatory

A number which represents something and which 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:

```
Resources [] :
```

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`, ...).

Source:

Value Range: <oi4Identifier>

Type: String

Access: Read only

Requirement: Mandatory

The `Source` (8.1.6) contains an `Oi4Identifier` (3), which identifies the asset (might be a device or an application) and where the `DataSet` comes from.

`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 `Source` must be unique within the scope of a publisher. Several `Resources` such as `MAM` or `Health` does not make usage of `Filter`.

NOTE For the `Resource` event, the `Filter` contains the associated event level.

NOTE The `Filter` shall be a topic encoded string as explained in 8.1.7.

`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.

`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 OI4 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>/<Source>/<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.

SOURCE_3:

With **SOURCE_3** the **DataSetMessage** will be published with other **DataSetMessages** of the same resource (8.1.5), same source (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_SOURCE_5`:

Enables publishing via the modes `APPLICATION_2` and `SOURCE_3`.

`APPLICATION_FILTER_6`:

Enables publishing via the modes `APPLICATION_2` and `FILTER_4`.

`SOURCE_FILTER_7`:

Enables publishing via the modes `SOURCE_3` and `FILTER_4`.

`APPLICATION_SOURCE_FILTER_8`:

Enables publishing via the modes `APPLICATION_2`, `SOURCE_3` and `FILTER_4`.

`ON_REQUEST_FILTER_9`

Enables publishing via the modes `ON_REQUEST_1` and `FILTER_4`.

`ON_REQUEST_SOURCE_10`

Enables publishing via the modes `ON_REQUEST_1` and `SOURCE_3`.

`ON_REQUEST_APPLICATION_11`

Enables publishing via the modes `ON_REQUEST_1` and `APPLICATION_2`.

`ON_REQUEST_SOURCE_FILTER_12`

Enables publishing via the modes `ON_REQUEST_1`, `SOURCE_3` and `FILTER_4`.

`ON_REQUEST_APPLICATION_FILTER_13`

Enables publishing via the modes `ON_REQUEST_1`, `APPLICATION_2` and `FILTER_4`.

`ON_REQUEST_APPLICATION_SOURCE_14`

Enables publishing via the modes `ON_REQUEST_1`, `APPLICATION_2` and `SOURCE_3`.

`ON_REQUEST_APPLICATION_SOURCE_FILTER_15`

Enables publishing via the modes `ON_REQUEST_1`, `APPLICATION_2`, `SOURCE_3` and `FILTER_4`.

NOTE *Be aware, Metadata is not a `DataSetMessage`, but a `DataSetMetaDataType`.*

Therefore, only `ON_REQUEST_FILTER_9` 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.

NOTE In case a `Get` request arrives over the Message Bus, the request will be handled regardless to `Interval` settings.

Precisions:

Value Range: `<PrecisionObject>`

Type: `Object`

Access: Read/Write

Requirement: Optional (all `Precisions = 0` (publish on change) if not present)

Each entry of the `Precisions` object 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 field>`:

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.

NOTE In case a `Get` request arrives over the Message Bus, the request will be handled regardless to `Precisions` settings.

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, it 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. Included are a topic path, an interval for application internal usage, and a configurability statement.

Because `SubscriptionList` is defined and referenced by this guideline, the OI4 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`: Freely configurable (delete is possible).

9.3.13 Interfaces

ATTENTION Information in this chapter requires alignment with other work 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 and LEDs for a single device.

Because `Interfaces` is defined and referenced by this guideline, the OI4 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.

NOTE The `ReferenceDesignation` is most likely set from outside the device or application. Therefore, it must be persisted on an application-specific basis.

9.3.15 Pagination

The Open Industry 4.0 Alliance' payload format for any kind of data are so called OPC UA PubSub `DataSets`. Each `DataSet` is packed in a single `DataSetMessage` (see 9.2.3). Several of this messages can be packed to send over the wire in a `NetworkMessage` (9.2.1).

Under these circumstances, a `NetworkMessage` may become excessively long. To prevent this, an optional `DataSetMessage` with `Pagination` information is available to indicate that there are other `NetworkMessages` with associated `DataSetMessages`.

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 which 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 which wants to know something and asks in that regard with `.../Get/<Resource>` can add a `PaginationRequest` object to its payload to define how many `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 to publish the information with use of `PaginationRequest`.

In general, `Pagination` information can be used in any context, for every `Method` and in combination with every `Resources`.

The `Pagination` object and `PaginationRequest` object differ for the methods `Pub` and `Get`. The differences are explained in [9.3.15.1](#) for `PaginationRequest` and in [9.3.15.2](#) for `Pagination`.

ATTENTION *If `NetworkMessages` has to be paginated, the provided `DataSetMessages` shall be fetched in a way, that they are consistent.*

An example of its use is explained in appendix [B1.1.15](#).

9.3.15.1 PaginationRequest

The payload for the method `Get` of a `DataSetMessage` ([9.2.3](#)) containing `PaginationRequest` information is structured as followed:

`PerPage`:

Value Range: `<number>`

Type: `UInt32`

Access: Read only

Requirement: Mandatory

A published message to the topic method `Get` might use `PerPage` to announce, how many `DataSetMessages` it expects maximum in the related publication.

NOTE *The value of `PerPage` must be > 0 .*

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 requested.

NOTE The value of `Page` must be > 0 to get a defined page with related information.

NOTE The value of `Page` must be 0 to request all information in a fragmented way with a maximal count of `PerPage DataSetMessages` for each fragment.

An example of its use is explained in appendix [B1.1.15](#).

9.3.15.2 Pagination

The payload for the methods `Pub` 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` 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 `NetworkMessage` 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` might use `PerPage` to announce, how many of the `TotalCount` `DataSetMessages` will be included in this 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.

`HasNext`:

Value Range: `<true or false>`

Type: Boolean

Access: Read only

Requirement: Mandatory

`HasNext` shows if an additional `NetworkMessage` can be expected.

`PaginationId`:

Value Range: `<MessageId>`

Type: String

Requirement: Mandatory

The `PaginationId` contains the `<MessageId>` of the first/initial paginated `NetworkMessage` in this series.

NOTE *To avoid misleading information because of messages received in the wrong order, the `PaginationId` can be used to restore the order.*

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 in the application, the following `Pub` message, containing the `CorrelationId` of the `Get` message, 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 like `"en-US"` or `"de-DE"` (see OPC UA [Part 3-8.4](#)) 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 OI4 Alliance

This chapter explains the most necessary methods and related data types, which the OI4 Alliance is using for call/reply pattern over the Message Bus.

All objects are JSON encoded.

9.4.1 FileUpload

ATTENTION *Information in this chapter requires alignment with other work 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](#).

`FileUpload` is a common service, globally defined by the OI4 Alliance. Because of that, 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 work 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](#).

`FileDownload` is a common service, globally defined by the OI4 Alliance. Because of that, a `DataSetClassId`, used by the `ServiceNetworkMessage` ([9.2.16](#)), is available for this service ([A2](#)).

9.4.3 FirmwareUpdate

ATTENTION *Information in this chapter requires alignment with other work 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](#).

`FirmwareUpdate` is a common service, globally defined by the OI4 Alliance. Because of that, a `DataSetClassId`, used by the `ServiceNetworkMessage` ([9.2.16](#)), is available for this service ([A2](#)).

9.4.4 Blink

ATTENTION *Information in this chapter requires alignment with other work 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](#).

`Blink` is a common service, globally defined by the OI4 Alliance. Because of that, a `DataSetClassId`, used by the `ServiceNetworkMessage` (9.2.16), is available for this service (A2).

9.4.5 `NewDataSetWriterId`

In some cases, it is required to set a new `DataSetMessage` in an application from the outside. For example, a reference designation (10.1.14), known by a MES shall be set into an OT connector. Therefore the IT connector, dealing with the MES system, has to set/create the `ReferenceDesignation` inside the OT connector. To do so, a valid and unique `DataSetWriterId` for the OT connector 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.

`NewDataSetWriterId` is a common service, globally defined by the OI4 Alliance. Because of that, 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 is 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 is 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 by 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` in seconds. Initial usage of the provided `DataSetWriterId` is only possible during this period of time.

10 Message Bus Communication

Chapter 8 discussed the structure of the topic, while in chapter 9, the data formats of the payload were described. This chapter addresses the use of both in combination.

Because the 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 the 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 in figure 16:

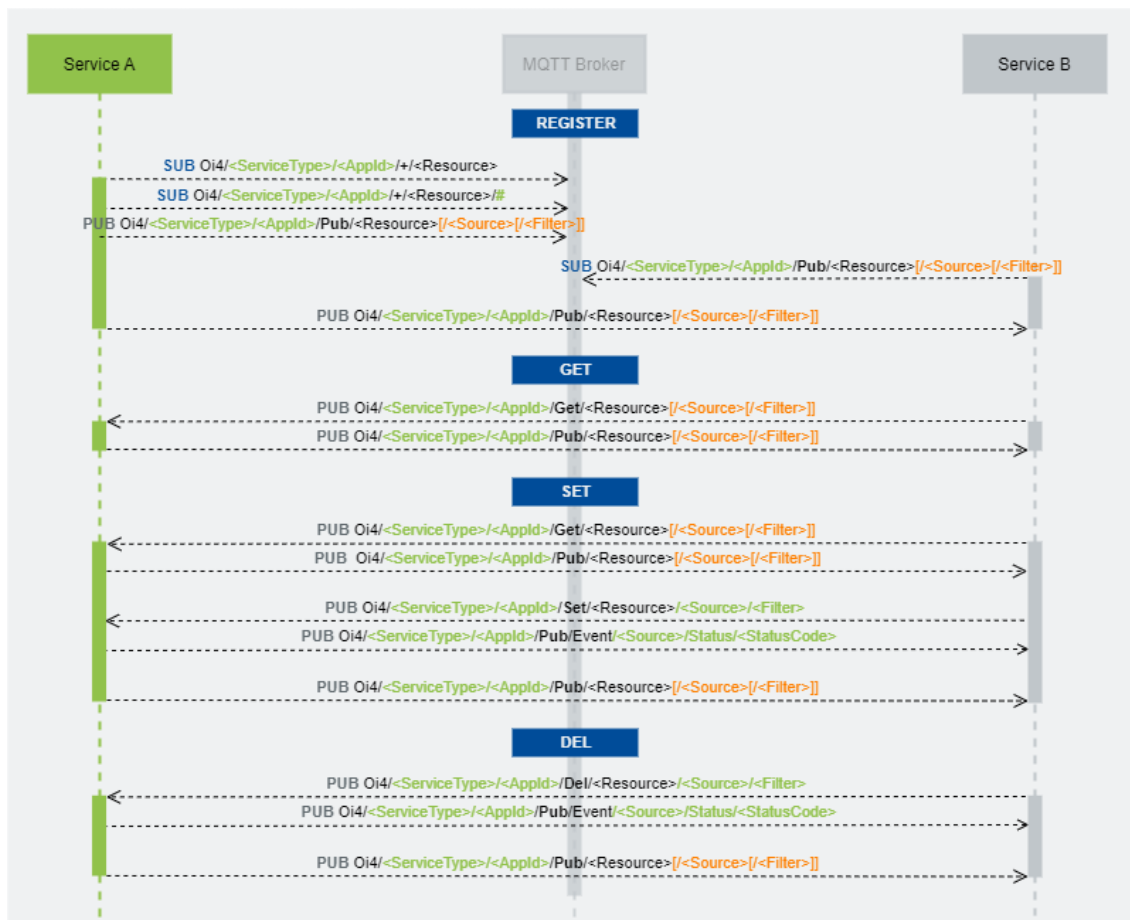


Figure 16: Sequence diagram of the four base interactions on how to use resources over the Message Bus

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 subscribing to the topics is done, *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* passes 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 content at the resource 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 successful, *Service A* must publish the changed resource on the `Pub` topic in regard 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 topic of the resource. 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 it must publish the changed resource on the `Pub` topic regarding to the settings in `PublicationList`.

The payload and the topic are related via `Source` and `Filter` of the `DataSetMessage`, contained in the payload, and `Source` and `Filter` contained in the topic.

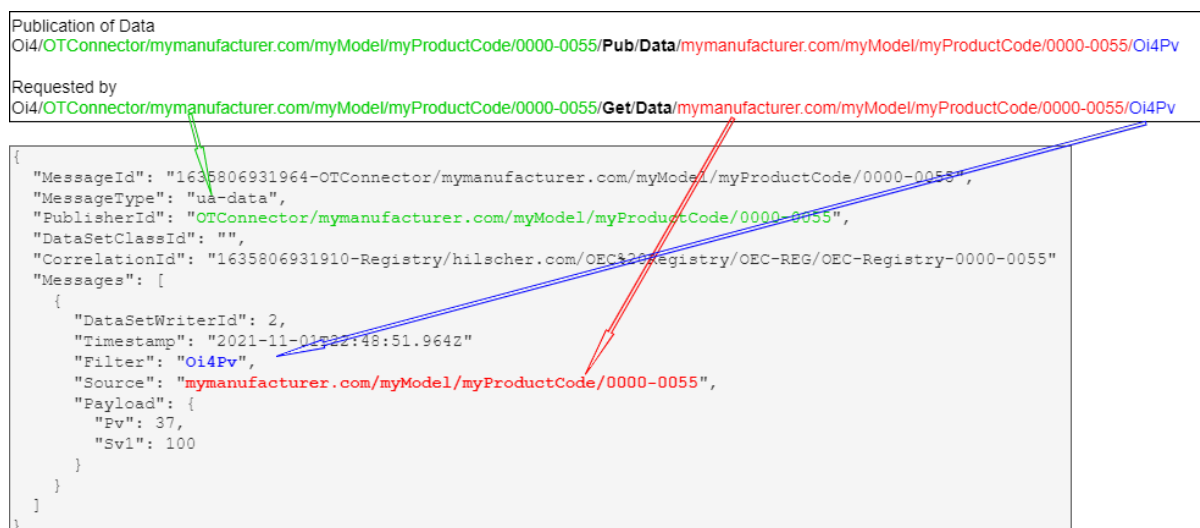


Figure 17: Relationship between topic and payload in standard pub/sub pattern

The following sub-chapters will explain how to use the available resources in combination with the related methods.

By using the methods `Set` and `Del`, 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 OI4 Alliance's predefined resources such as `MAM`, `Health`, `Config` and so on.

The metadata for all other resources than `Data` must be specified by the Open Industry 4.0 Alliance. The OI4 Alliance must provide the definitions and the related `DataSetClassId` for them.

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 chapter [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 `Source` can be used to request only `MAM` information of a specific asset. Without `Source`, all `MAM` information will be requested.
- In combination with the method `Pub`, the payload contains `DataSetMessages` of type `MAM` and optional `Pagination` ([9.3.15](#)).

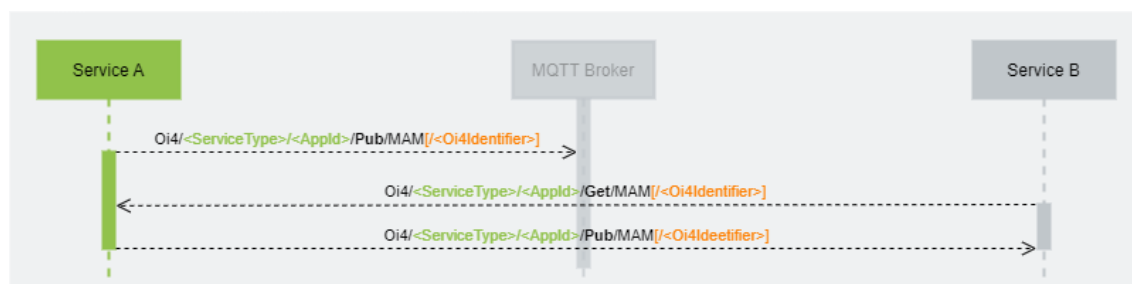


Figure 18: Sequence diagram of the two basic interactions that can be used for the resource MAM

Get MAM from assets

Via the `<Method>/<Resource>[/<Source>]` 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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/MAM`.

Publish MAM from assets

The information is provided via the `<Method>/<Resource>[/<Source>]` 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>/<ApplId>",
  "DataSetClassId": "360ca8f3-5e66-42a2-8f10-9cdf45f4bf58",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "",
      "Source": "<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 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 Source can be used to request only Health information of a specific asset. Without Source, 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).

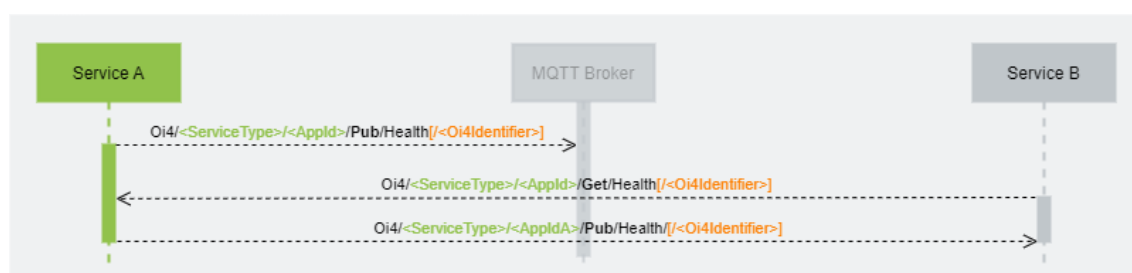


Figure 19: Sequence diagram of the two basic interactions that can be used for the resource Health

Get Health from assets

Via the <Method>/<Resource> [/<Source>] 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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/Health`.

Publish Health from assets

The information is provided via the `<Method>/<Resource>[/<Source>]` 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": "",
      "Source": "<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 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 `Source` can be used to request only `Config` objects of a specific asset. The `Filter` can be used to request a specific `Config` object. Without `Source` 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`, `Source` and `Filter` to be valid.

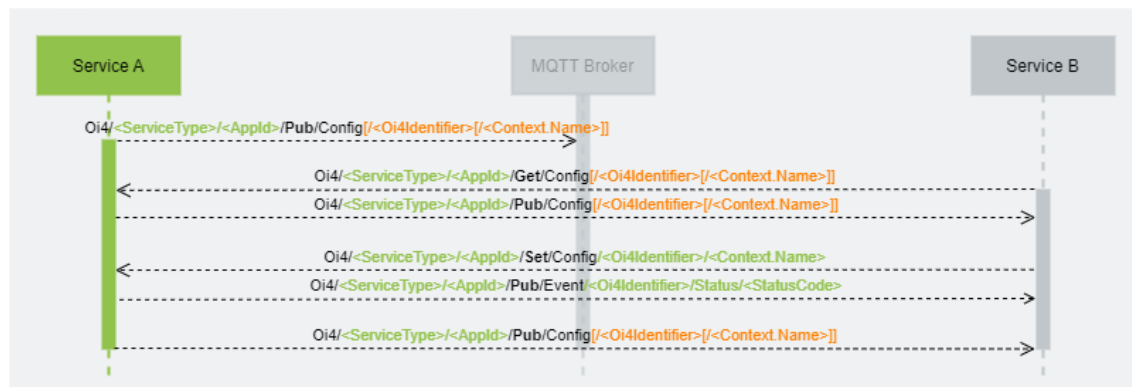


Figure 20: Sequence diagram of the three basic interactions that can be used for the resource `Config`

Get Config from assets

Via the `<Method>/<Resource>[/<Source>[/<Filter>]]` combination `Get/Config[/<Oi4Identifier>/<Context.Name>]]` it is possible to trigger a (re-)publishing of this information to the `Pub/Config[/<Oi4Identifier>/<Context.Name>]]` topic.

Publisher topic:

Get explicit `DataSetMessage` for a specific `Config`:

`Oi4/<ServiceType>/<AppId>/Get/Config/<Oi4Identifier>/<Context.Name>`

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.*


```

{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<ApplId>",
  "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>",
      "Source": "<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)

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<Method>/<Resource>/<Source>/<Filter>` combination

`Set/Config/<Oi4Identifier>/<Context.Name>` 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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Set` request.

NOTE *Each publication, using the method `Set` must be done explicitly. That means `Set/Config` is only possible by using a complete topic containing `Resource`, `Source` 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[/<Source>[/<Filter>]]` topic, according to the settings in `PublicationList` (9.3.11).

Publisher topic:

Set explicit `DataSetMessage` for a specific `Config`:

`Oi4/<ServiceType>/<AppId>/Get/Config/<Oi4Identifier>/<Context.Name>`

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>",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "<ConfigNameA>": {
          "Value": "<new value>"
        },
        "<ConfigNameB>": {
          "Value": "<new value>"
        }
      }
    }
  ]
}
```



```

{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<ApplId>",
  "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>",
      "Source": "<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)

RECOMMENDED *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, it simply requires a `Set` with empty value of the config object.

```

{
  "MessageId": "<unixTimestampInMs-PublisherId>",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<ApplId>",
  "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>",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "<ConfigNameA>": {
          "Value": ""
        },
        "<ConfigNameB>": {
          "Value": ""
        }
      }
    }
  ]
}

```

NOTE More 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 `Source` can be used to request only `License` information of a

specific asset. Without `Source`, all `License` information will be requested. Additional to the `Source`, 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).

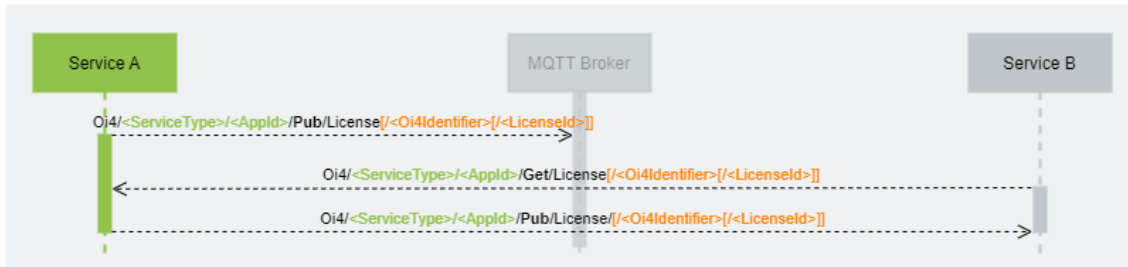


Figure 21: Sequence diagram of the two basic interactions that can be used for the resource `License`

Get License from assets

Via the `<Method>/<Resource>[/<Source>[/<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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/License`.

Publish License from assets

The information is provided via the `<Method>/<Resource>[/<Source>[/<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>/<Appld>",
  "DataSetClassId": "2ae0505e-2830-4980-b65e-0bbdf08e2d45",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "Apache%202.0",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Components": [
          {
            "Component": "MQTTCLI",
            "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 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 `Source` can be used to request only `LicenseText` information of a specific asset. Without `Source`, all `LicenseText` information will be requested. Additional to the `Source`, 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 `DataSetMessages` of type `LicenseText` and optional `Pagination` ([9.3.15](#)).

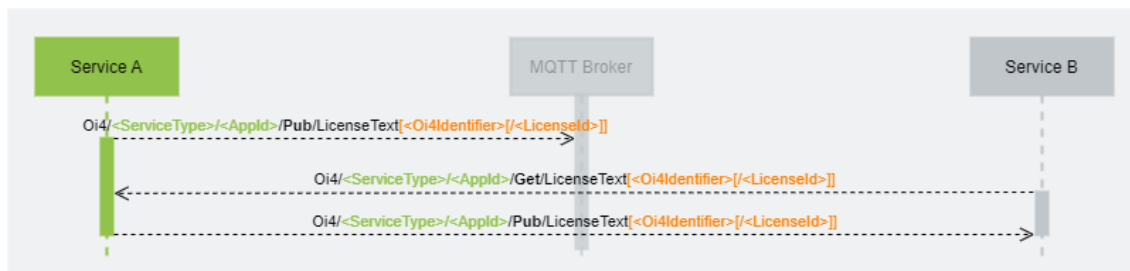


Figure 22: Sequence diagram of the two basic interactions that can be used for the resource `LicenseText`

Get LicenseText from assets

Via the `<Method>/<Resource>[/<Source>[/<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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/LicenseText`.

Publish LicenseText from assets

The information is provided via the `<Method>/<Resource>[/<Source>[/<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>",
      "Source": "<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 examples can be found in the appendix B1.1.5.*

10.1.6 RtLicense

ATTENTION *Information in this chapter requires alignment with other work 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.

- 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 `RtLicense` object is not included. The `Source` can be used to request only `RtLicense` information of a specific asset. Without `Source`, all `RtLicense` information will be requested.
- In combination with the method `Pub`, the payload contains `DataSetMessages` of type `RtLicense` and optional `Pagination` (9.3.15).

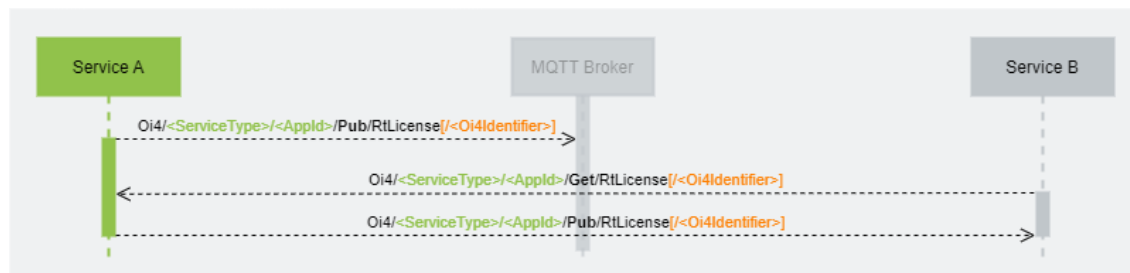


Figure 23: Sequence diagram of the two basic interactions that can be used for the resource `RtLicense`

Get `RtLicense` from assets

Via the `<Method>/<Resource> [/<Source>]` 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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/RtLicense`.

Publish RtLicense from assets

The information is provided via the `<Method>/<Resource>[/<Source>]` 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>/<ApplId>",
  "DataSetClassId": "ebd12d4b-da1c-4671-ab86-db102fecc603",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "",
      "Source": "<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 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 OI4 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 OI4 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 Oi4Pv 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 `Source` can be used to request only `Data` objects of a specific asset. The `Filter` can be used to request a specific `<Tag>`. Without `Source` 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`, `Source` and `Filter` to be valid.

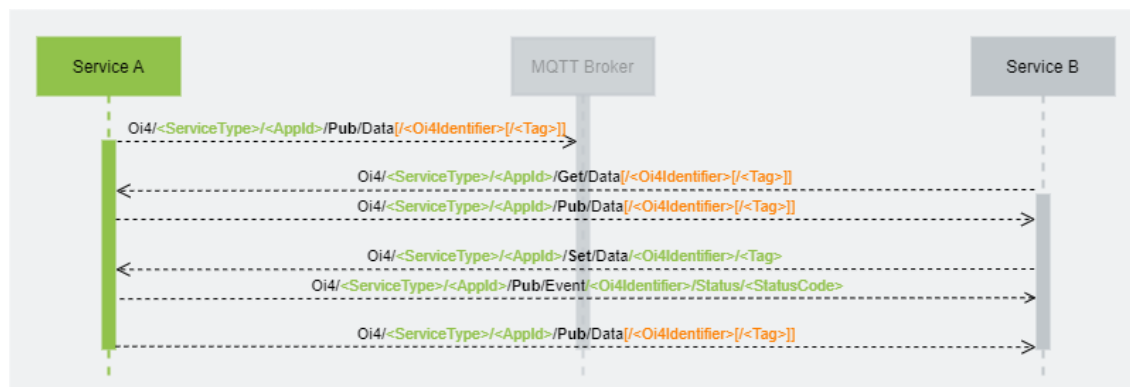


Figure 24: Sequence diagram of the three basic interactions that can be used for the resource `Data`

Get Data from assets

Via the `<Method>/<Resource>[/<Source>[/<Filter>]]` combination `Get/Data[/<Oi4Identifier>[/<Tag>]]`, the publication of this information to the `Pub/Data[/<Oi4Identifier>[/<Tag>]]` topic can be triggered.

Publisher topic:

Get explicit `DataSetMessage` for a specific `Tag`:

`Oi4/<ServiceType>/<AppId>/Get/Data/<Oi4Identifier>/<Tag>`

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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/Data`.

Publish Data from assets

The information is provided via the `<Method>/<Resource>[/<Source>[/<Filter>]]` combination `Pub/Data[/<Oi4Identifier>[/<Tag>]]`.

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 `Tag`:

`Oi4/<ServiceType>/<AppId>/Pub/Data/<Oi4Identifier>/<Tag>`

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>/<ApplId>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "<filter>",
      "Source": "<Oi4Identifier>",
      "SequenceNumber": <UINT32>,
      "MetaDataVersion": {
        "MajorVersion": <UINT32>,
        "MinorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Status": <UINT32>,
      "Payload": {
        <DataSet>
      }
    }
  ]
}
```


Payload if `Filter` equals `Oi4Pv`:

```

{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<ApplId>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "Oi4Pv",
      "Source": "<Oi4Identifier>",
      "SequenceNumber": <UINT32>,
      "MetaDataVersion": {
        "MajorVersion": <UINT32>,
        "MinorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Status": <UINT32>,
      "Payload": {
        "Pv": <e.g. subset of OPC UA Base Types>,
        "Sv1": {<e.g. complex JSON object>},
        "Sv2": [<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

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<Method>/<Resource>/<Source>/<Filter>` combination

`Set/Data/<Oi4Identifier>/<Tag>` 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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Set` request.

NOTE Each publication, using the method `Set` must be done explicitly. That means `Set/Data` is only possible by using a complete topic containing `Resource`, `Source` 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[/<Source>/<Filter>]` topic, according to the settings in `PublicationList` (9.3.11).

Publisher topic:

Set explicit `DataSetMessage` for a specific `Tag`:

`Oi4/<ServiceType>/<AppId>/Set/Data/<Oi4Identifier>/<Tag>`

Payload for asset defined `Filter`:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<ApplId_of_DataSetOwner>",
  "DataSetClassId": "<(meta)data specific GUID>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "<filter>",
      "Source": "<Oi4Identifier>",
      "MetaDataVersion": {
        "MajorVersion": <UINT32>,
        "MinorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Payload": {
        <DataSet>
      }
    }
  ]
}
```

Payload if `Filter` equals `Oi4Pv`:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<Appld_of_DataSetOwner>",
  "DataSetClassId": "<(meta)data specific GUID>",
  "correlationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "Oi4Pv",
      "Source": "<Oi4Identifier>",
      "MetaDataVersion": {
        "MajorVersion": <UINT32>,
        "MinorVersion": <UINT32>
      },
      "Timestamp": "<DateTime>",
      "Payload": {
        "Pv": <e.g. subset of OPC UA Base Types>,
        "Sv1": {<e.g. complex JSON object>},
        "Sv2": [<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 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 OI4 Alliance 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`, `Source` 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`, `Source` and `Filter` to be valid.

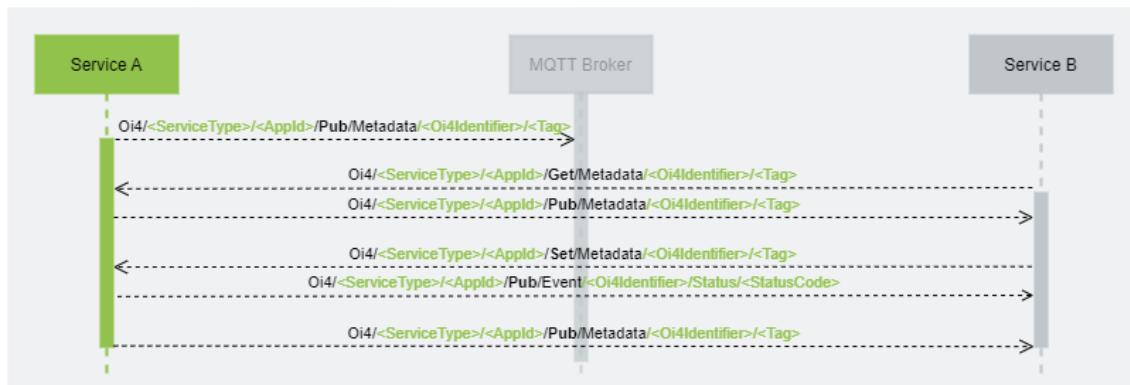


Figure 25: Sequence diagram of the three basic interactions that can be used for the resource `Metadata`

Get Metadata from specific asset/filter

Via the `<Method>/<Resource>/<Source>/<Filter>` combination `Get/Metadata/<Oi4Identifier>/<Tag>`, the publication of this information to the `Pub/Metadata/<Oi4Identifier>/<Tag>` topic can be triggered.

Publisher topic:

Get explicit `DataSetMetaData` for a specific `Tag`:

`OI4/<ServiceType>/<AppId>/Get/Metadata/<Oi4Identifier>/<Tag>`

Payload:


```

{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-metadata",
  "PublisherId": "<ServiceType>/<AppId>",
  "DataSetWriterId": <UINT16>,
  "Filter": "<Filter>",
  "Source": "<Oi4Identifier>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "MetaData": {
    <DataSetMetaDataType>
  }
}

```

Set Metadata from specific asset/filter

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<Method>/<Resource>/<Source>/<Filter>` combination `Set/Metadata/<Oi4Identifier>/<Tag>` 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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Set` request.

NOTE *Each publication using the method `Set` must be done explicitly. That means `Set/Metadata` is only possible by using a complete topic containing `Resource`, `Source` and `Filter`, such as `Oi4/<ServiceType>/<AppId>/Set/Metadata/<Oi4Identifier>/<Tag>`.*

If setting `Metadata` was successful, the new `Metadata` will be published to the `Pub/Metadata/<Oi4Identifier>/<Tag>` topic, according to the settings in `PublicationList` (9.3.11).

Publisher topic:

Set explicit `DataSetMetaData` for a specific `Tag`:
`Oi4/<ServiceType>/<AppId>/Get/Metadata/<Oi4Identifier>/<Tag>`

Payload:

```

{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-metadata",
  "PublisherId": "<ServiceType>/<Appld_of_MetadataOwner>",
  "DataSetWriterId": <UINT16>,
  "Filter": "<Filter>",
  "Source": "<Oi4Identifier>",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "MetaData": {
    <DataSetMetaDataType>
  }
}

```

NOTE More 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 `Filter`, as part of the topic, contains information about the `EventCategory` as well as `EventLevel` - separated with a forward slash. A topic without both of this information 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.

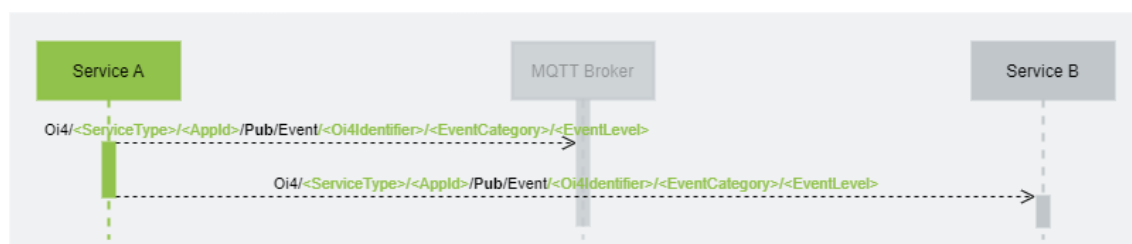


Figure 26: Sequence diagram of the basic interaction that can be used for the resource `Event`

Publish event

The information is provided by the `<Method>/<Resource>/<Source>/<Filter>` combination `Pub/Event/<Oi4Identifier>/<EventCategory>/<EventLevel>`.

Applications can listen to any new event by subscribing to the topic

`Oi4/+/+/+/+/+/Pub/Event/#`.

Applications can filter for all events of a specific asset (application or device) by subscribing to the topic `Oi4/+/+/+/+/+/Pub/Event/<Oi4Identifier>/#`.

Applications can filter for all events of a specific category by subscribing to the topic

`Oi4/+/+/+/+/+/Pub/Event/+/+/+/+/<EventCategory>/#`.

Applications can filter for all events of a specific category and level by subscribing to the topic

`Oi4/+/+/+/+/+/Pub/Event/+/+/+/+/<EventCategory>/<EventLevel>`.

Publisher topic:

Explicit `DataSetMessage` for a single `EventLevel`:

`Oi4/<ServiceType>/<AppId>/Pub/Event/<Oi4Identifier>/<EventCategory>/<EventLevel>`

Payload:

```

{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<Appld>",
  "DataSetClassId": "543ae05e-b6d9-4161-a0a3-350a0fac5976",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": " <EventCategory>/<EventLevel>",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Number": <UINT32>,
        "Description": "<string>",
        "Category": "<category>"
        "Details": {
          <depending on the event category>
        }
      }
    }
  ]
}

```

NOTE Only DataSetMessages of type Event with same EventCategory and EventLevel can be combined in a single NetworkMessage.

Events are divided into the following EventCategories, depending on the source:

- Syslog (see [9.3.9.2](#))
- 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 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 Filter for this category are described in [8.1.7/Ne107](#)
- Category in payload is CAT_NE107_2

- Generic ([9.3.9.4](#))
- Topic `Filter` for this category are described in [8.1.7/Generic](#)
- Category in payload is `CAT_GENERIC_99`

NOTE *More 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 `Source` can be used to request only `Profile` information of a specific asset. Without `Source`, all `Profile` information will be requested.
- In combination with the method `Pub`, the payload contains `DataSetMessages` of type `Profile` and optional `Pagination` (9.3.15).

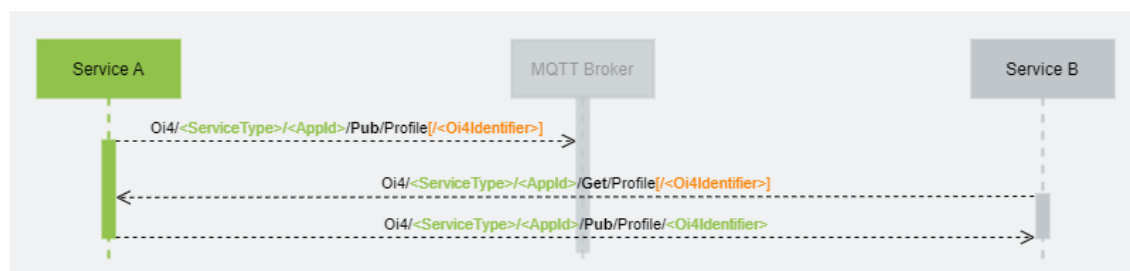


Figure 27: Sequence diagram of the two basic interactions that can be used for the resource `Profile`

Get Profile from assets

Via the `<Method>/<Resource> [/<Source>]` 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 single `NetworkMessage` and published to the topic `Oi4/<ServiceType>/<AppId>/Pub/Profile`.

Publish Profile from assets

The information is provided via the `<Method>/<Resource>[/<Source>]` 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": "",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Resources": [
          "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 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`, `Source`, `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 `Source` 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 `Source` 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 `DataSetMessages` 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`, `Source` 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`.

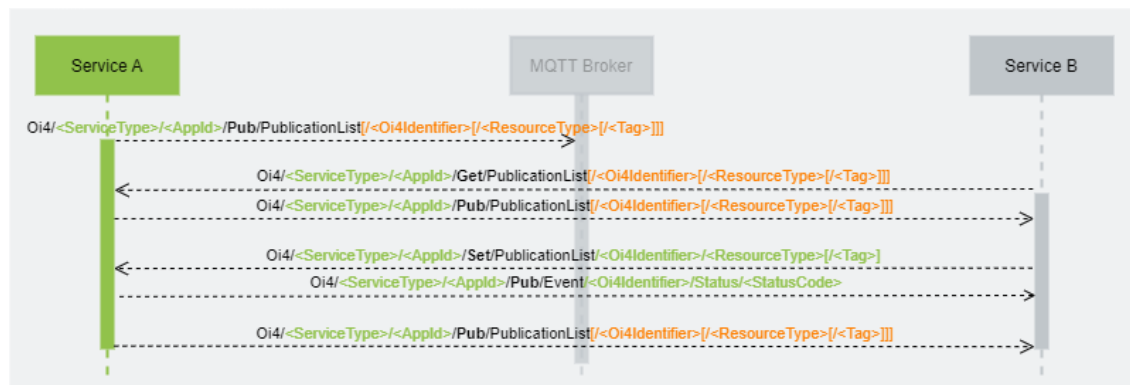


Figure 28: Sequence diagram of the three basic interactions that can be used for the resource `PublicationList`

Get `PublicationList` from assets

Via the `<Method>/<Resource>[/<Source>[/<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 single `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>/<Source>[/<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>/<Appld>",
  "DataSetClassId": "217434d6-6e1e-4230-b907-f52bc9ffe152",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "<ResourceType>",
      "Source": "<Oi4Identifier = Appld>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Resource": "<ResourceType>",
        "Source": "<Oi4Identifier>",
        "Filter": "<Tag>",
        "DataSetWriterId": <UINT16>,
        "Mode": "<<EnumName>_<EnumValue>>",
        "Interval": <UINT32>,
        "Precisions": {
          "<Key m of DataSet>": <REAL>,
          "<Key n of DataSet>": <REAL>
        },
        "Config": "<<EnumName>_<EnumValue>>"
      }
    },
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "MAM",
      "Source": "<Oi4Identifier = Appld>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Resource": "MAM",
        "Source": "<oi4Identifier>",
        "Filter": "",
        "DataSetWriterId": <UINT16>,
        "Mode": "<<EnumName>_<EnumValue>>",
        "Interval": <UINT32>,
        "Precisions": {

```

```

    "<Key m of DataSet>": <REAL>,
    "<Key n of DataSet>": <REAL>
  },
  "Config": "<<EnumName>_<EnumValue>>"
}
},
{
  "DataSetWriterId": <UINT16>,
  "Filter": "Health",
  "Source": "<Oi4Identifier = AppId>",
  "Timestamp": "<DateTime>",
  "Payload": {
    "Resource": "Health",
    "Source": "<Oi4Identifier>",
    "Filter": "",
    "DataSetWriterId": <UINT16>,
    "Mode": "<<EnumName>_<EnumValue>>",
    "Interval": <UINT32>,
    "Precisions": {
      "<Key m of DataSet>": <REAL>,
      "<Key n of DataSet>": <REAL>
    },
    "Config": "<<EnumName>_<EnumValue>>"
  }
},
{
  "DataSetWriterId": <UINT16>,
  "Filter": "Event",
  "Source": "<Oi4Identifier = AppId>"
  "Timestamp": "<DateTime>",
  "Payload": {
    "Resource": "Event",
    "Source": "<Oi4Identifier>",
    "Filter": "<EventCategory>/<EventLevel>",
    "DataSetWriterId": <UINT16>,
    "Mode": "<<EnumName>_<EnumValue>>",
    "Interval": <UINT32>,
    "Precisions": {
      "<Key m of DataSet>": <REAL>,

```

```

    "<Key n of DataSet>": <REAL>
  },
  "Config": "<<EnumName>_<EnumValue>>"
}
},
{
  "DataSetWriterId": <UINT16>,
  "Filter": "Data",
  "Source": "<Oi4Identifier = Appld>",
  "Timestamp": "<DateTime>",
  "Payload": {
    "Resource": "Data",
    "Source": "<Oi4Identifier>",
    "Filter": "<Tag>",
    "DataSetWriterId": <UINT16>,
    "Mode": "<<EnumName>_<EnumValue>>",
    "Interval": <UINT32>,
    "Precisions": {
      "<Key m of DataSet>": <REAL>,
      "<Key n of DataSet>": <REAL>
    },
    "Config": "<<EnumName>_<EnumValue>>"
  }
}
]
}

```

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

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<Method>/<Resource>/<Source>/<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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Set` request.

NOTE Each publication, using the method `Set`, must be done explicitly. That means `Set/PublicationList` is only possible by using a complete topic containing `Resource`, `Source` 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 to 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>]`

```
{
  "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>",
      "Source": "<Oi4Identifier = AppId>",
      "Timestamp": "<DateTime>",
      "Payload": {
        {
```

```

    "Resource": "<ResourceType>",
    "Source": "<Oi4Identifier>",
    "Filter": "<Tag>",
    "DataSetWriterId": <UINT16>,
    "Mode": "<<EnumName>_<EnumValue>>",
    "Interval": <UINT32>,
    "Precisions": {
      "<Key m of DataSet>": <REAL>,
      "<Key n of DataSet>": <REAL>
    },
    "Config": "<<EnumName>_<EnumValue>>"
  }
}
]
}

```

Payload:

NOTE More 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 `Source` 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 `Source` and `Filter`, all `SubscriptionList` 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 `DataSetMessages` 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`, `Source` 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`, `Source` 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`.

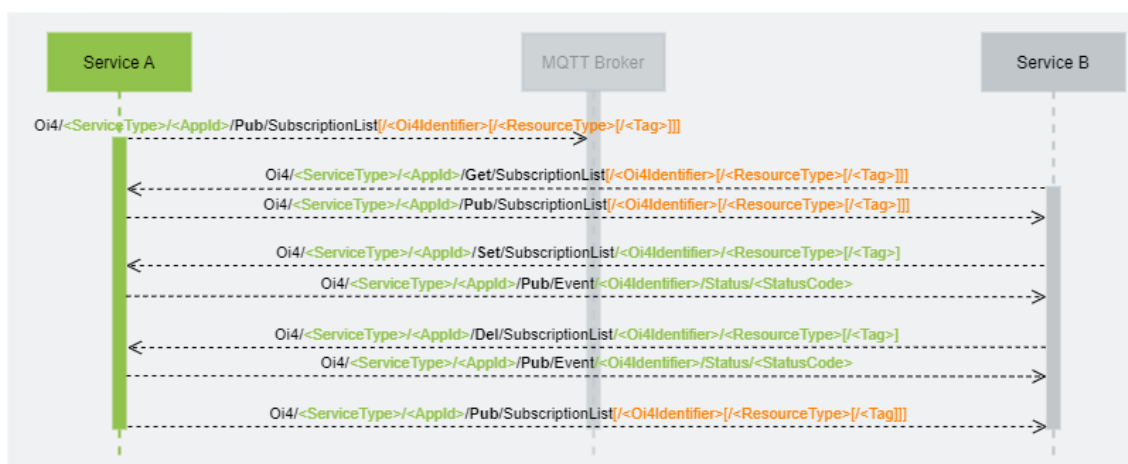


Figure 29: Sequence diagram of the four basic interactions that can be used for the resource `SubscriptionList`

Get SubscriptionList from assets

Via the `<Method>/<Resource>[/<Source>[/<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 single `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>/<Source>[/<Filter>]` combination `Pub/SubscriptionList/<Oi4Identifier>[/<ResourceType>[/<Tag>]]` might be used.

Any application can listen to any new `SubscriptionList` 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>/<Appld>",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "<ResourceType>",
      "Source": "<Oi4Identifier = Appld>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "TopicPath": "<Topic>",
        "Interval": <UINT32>,
        "Config": "<<EnumName>_<EnumValue>>"
      }
    }
  ]
}

```

NOTE In case a `SubscriptionList` objects contains wildcards in the `TopicPath`, it is in the responsibility of the application to evaluate which `SubscriptionList` entries matches the request and must be published.

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)

RECOMMENDED To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.

Via the `<Method>/<Resource>/<Source>/<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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Set` request.

NOTE *Each publication, using the method `Set`, must be done explicitly. That means, `Set/SubscriptionList` is only possible by using a complete topic containing `Resource`, `Source` 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 to 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>/<Tag>>",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "TopicPath": "<Topic>",
        "Interval": <UINT32>,
        "Config": "<<EnumName>_<EnumValue>>"
      }
    }
  ]
}

```

Delete SubscriptionList

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<Method>/<Resource>/<Source>/<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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Del` request.

NOTE *Each publication, using the method `Del` must be done explicitly. That means `Del/SubscriptionList` is only possible by using a complete topic containing `Resource`, `Source` 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>[/<Tag>]>"
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "TopicPath": "<Topic>",
        "Interval": <UINT32>,
        "Config": "<<EnumName>_<EnumValue>>"
      }
    }
  ]
}
```

NOTE More examples can be found in the appendix [B1.1.12](#).

10.1.13 Interfaces

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

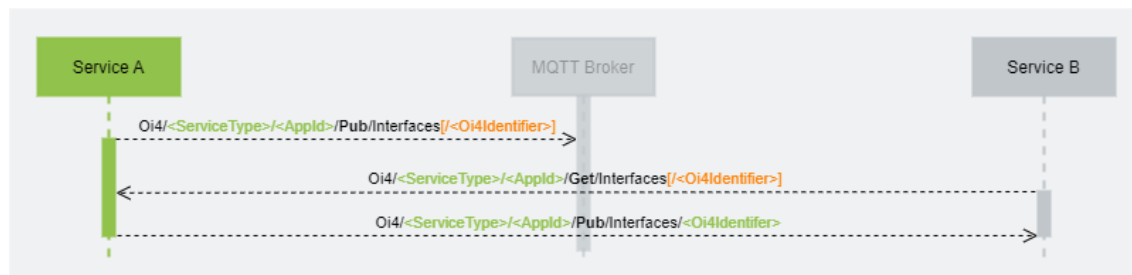


Figure 30: Sequence diagram of the two basic interactions that can be used for the resource Interfaces

NOTE More 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 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 `Source` can be used to request only the `ReferenceDesignation` objects of a specific asset. Without `Source`, all `ReferenceDesignation` objects will be requested.
- In combination with the method `Pub`, the payload contains `DataSetMessages` 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 `Source` 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 `Source` to be valid.



Figure 31: Sequence diagram of the interactions that can be used for the resource ReferenceDesignation

Get ReferenceDesignation from assets

Via the `<Method>/<Resource> [/<Source>]` 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 single 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>/<Source> 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/ReferenceDesignation/#.

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 splitted via pagination mechanism into several messages.

Payload:

```
{
  "MessageId": "unixTimestampInMs-PublisherId",
  "MessageType": "ua-data",
  "PublisherId": "<ServiceType>/<ApplId>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Location": {
          "Value": "<designation value>",
          "Local": "<local designation value>",
          "Parent": {
            "Value": "<designation value>",
            "Local": "<local designation value>",
            "Oi4Identifier": "<Oi4Identifier>"
          }
        }
      },
      "Function": {
        "Value": "<designation value>",
        "Local": "<local designation value>",
        "Parent": {
          "Value": "<designation value>",
          "Local": "<local designation value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      },
      "Product": {
        "Value": "<designation value>",
        "Local": "<local designation value>",
        "Parent": {
          "Value": "<designation value>",
          "Local": "<local designation value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      }
    }
  ]
}
```

```

    }
  }
}
]
}

```

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

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

Via the `<Method>/<Resource>/<SubResource>` 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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Set` request.

NOTE *Each publication, using the method `Set` must be done explicitly. That means `Set/ReferenceDesignation` is only possible by using a complete topic containing `Resource` and `Source`, 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>/<ApplId>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": [
    {
      "DataSetWriterId": <UINT16>,
      "Filter": "",
      "Source": "<Oi4Identifier>",
      "Timestamp": "<DateTime>",
      "Payload": {
        "Location": {
          "Value": "<designation value>",
          "Local": "<local designation value>",
          "Parent": {
            "Value": "<designation value>",
            "Local": "<local designation value>",
            "Oi4Identifier": "<Oi4Identifier>"
          }
        }
      },
      "Function": {
        "Value": "<designation value>",
        "Local": "<local designation value>",
        "Parent": {
          "Value": "<designation value>",
          "Local": "<local designation value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      },
      "Product": {
        "Value": "<designation value>",
        "Local": "<local designation value>",
        "Parent": {
          "Value": "<designation value>",
          "Local": "<local designation value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      }
    }
  ]
}

```

```

    }
  }
}
]
}

```

Delete ReferenceDesignation from specific asset

RECOMMENDED To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.

Via the `<Method>/<Resource>/<SubResource>` 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/<Oi4Identifier>/Status/<StatusCode>` (9.3.9.1) to provide feedback to the caller. The caller finds the related `Event` through the `CorrelationId`, which is equal to the `MessageId` of its `Del` request.

NOTE Each publication, using the method `Del` must be done explicitly. That means `Del/ReferenceDesignation` is only possible by using a complete topic containing `Resource` and `Source`, 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>/<Appld>",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "<empty/omitted> or <initial MessageId>",
  "Messages": []
}
    
```

NOTE More 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 OI4 Alliance-compliant application that supports these services.

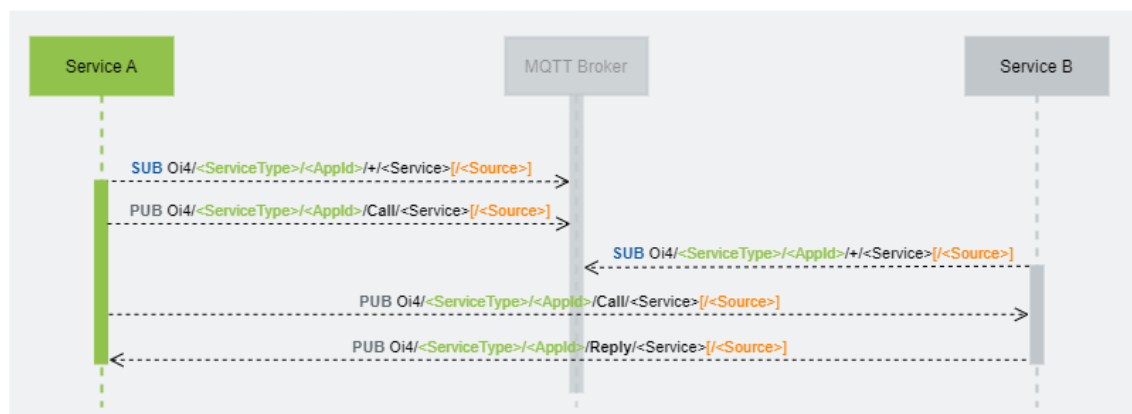


Figure 32: Sequence diagram of the interactions on how to use method calls over the Message Bus

10.2.1 FileUpload

ATTENTION Information in this chapter requires alignment with other work 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](#).

RECOMMENDED 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.

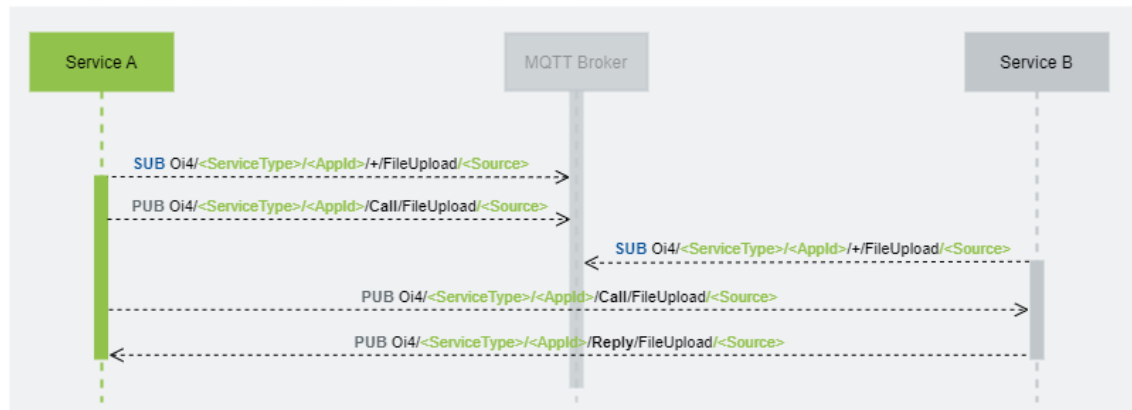


Figure 33: Sequence diagram of the interactions on how to use the method call FileUpload over the Message Bus

The information is provided via the `<Method>/<Services>/<Source>` 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/omitted> 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>/<appld>",
  "DataSetClassId": "3b4a62ba-026f-4ee8-bc99-3a5f85fc9f3b",
  "CorrelationId": "<Caller MessageId>",
  "Message": {
    "Results": [
      {
        "StatusCode": "<StatusCode>",
        "InputArgumentResults": ["<StatusCode>"],
        "OutputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}

```

10.2.2 FileDownload

ATTENTION *Information in this chapter requires alignment with other work 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](#).

RECOMMENDED *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.

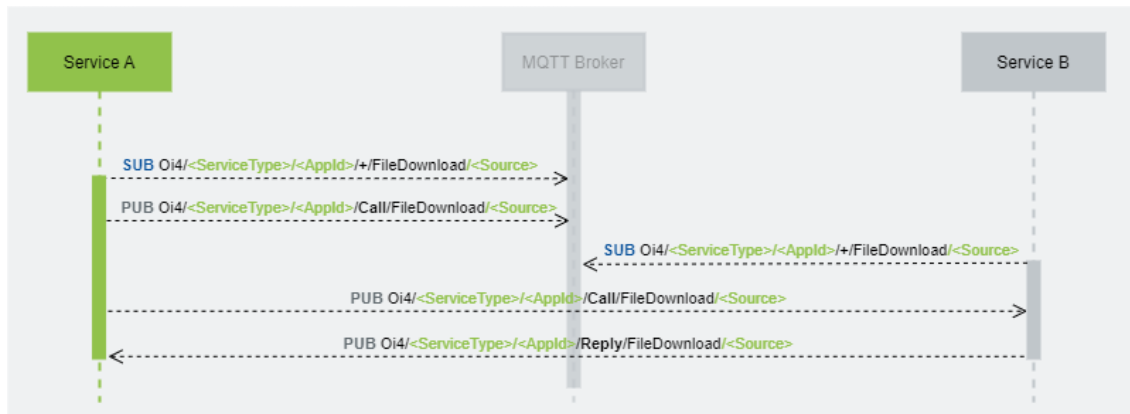


Figure 34: Sequence diagram of the interactions on how to use the method call FileDownload over the Message Bus

The information is provided via the `<Method>/<Services>/<Source>` 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/omitted> 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>/<ApplId>",
  "DataSetClassId": "760abda2-ba40-4e6e-863a-eea8c002b4e4",
  "CorrelationId": "<Caller MessageId>",
  "Message": {
    "Results": [
      {
        "StatusCode": "<StatusCode>",
        "InputArgumentResults": ["<StatusCode>"],
        "OutputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}

```

10.2.3 FirmwareUpdate

ATTENTION *Information in this chapter requires alignment with other work 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](#).

RECOMMENDED *To prevent write access by unauthorized communication participants, the operator must prevent this via ACL.*

This common function cannot check whether the firmware is valid or not. The legitimacy of the firmware must be checked by the application after receiving it. Additionally, technology dependent state machines for update procedures might be necessary.

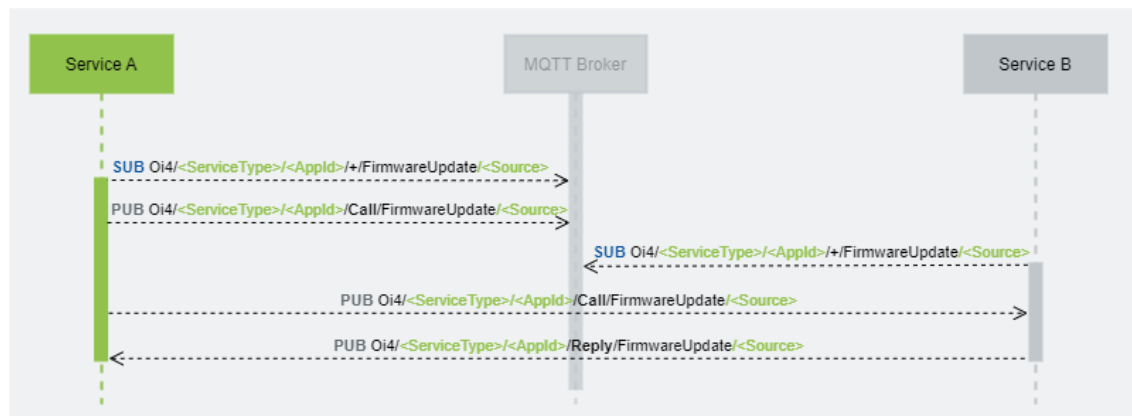


Figure 35: Sequence diagram of the interactions on how to use the method call FirmwareUpdate over the Message Bus

The information is provided via the `<Method>/<Services>/<Source>` 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/omitted> 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>",
        "InputArgumentResults": ["<StatusCode>"],
        "OutputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

10.2.4 Blink

ATTENTION *Information in this chapter requires alignment with other work 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.

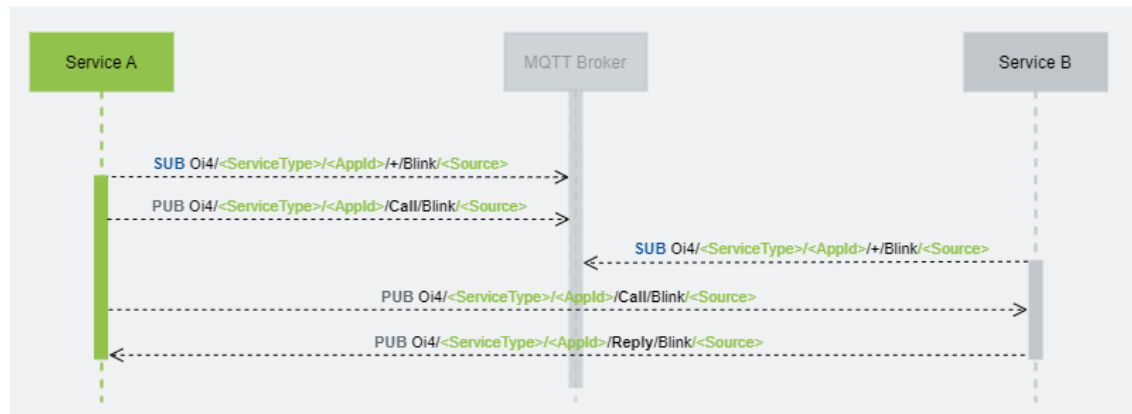


Figure 36: Sequence diagram of the interactions on how to use the method call Blink over the Message Bus

The information is provided via the `<Method>/<Services>/<Source>` 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/omitted> 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>",
        "InputArgumentResults": ["<StatusCode>"],
        "OutputArguments": [
          {
            <todo>
          }
        ]
      }
    ]
  }
}
```

10.2.5 NewDataSetWriterId

In some cases it is required to set a new `DataSetMessage` in an application from the outside of it, e.g., a reference designation (9.3.14), known by a MES system shall be written into an OT connector. Therefore the IT connector, dealing with the MES system, has to set/create the `ReferenceDesignation` inside the OT connector. To do so, a valid and unique `DataSetWriterId` for the OT connector 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.

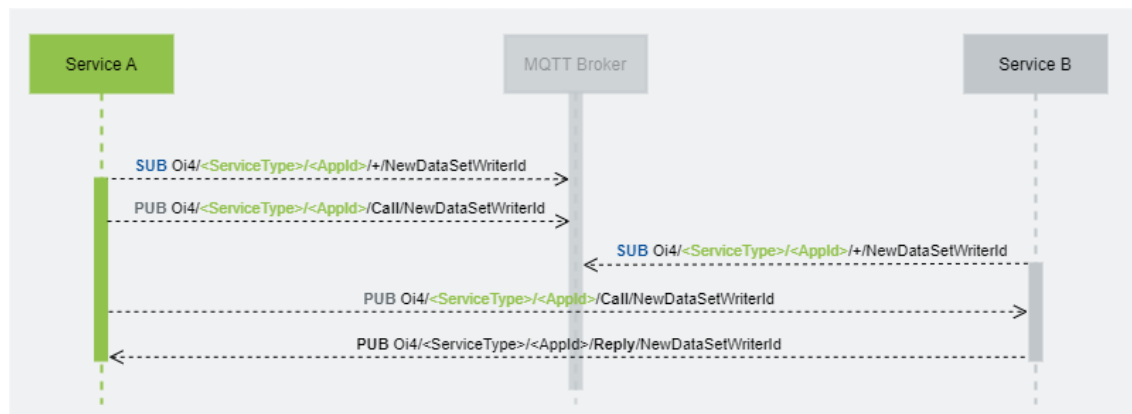


Figure 37: Sequence diagram of the interactions on how to use the method call `NewDataSetWriterId` over the Message Bus

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/omitted> 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>,
        "InputArgumentResults": [<UINT32>],
        "OutputArguments": [
          {
            "DataSetWriterId": <UINT16>,
            "Ttl": <UINT32>
          }
        ]
      }
    ]
  }
}
```

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 work groups and has not been maturely specified.*

The service `Read` returns the data, which were requested.

RECOMMENDED *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 work groups and has not been maturely specified.*

The service `Write` writes the given data to a defined asset.

RECOMMENDED *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 work 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 work 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 work 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.

RECOMMENDED *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 Open Industry 4.0 Alliance, are shown in [figure 38](#). They have typical characteristics:

- Consisting of at least one edge computer.
- 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.

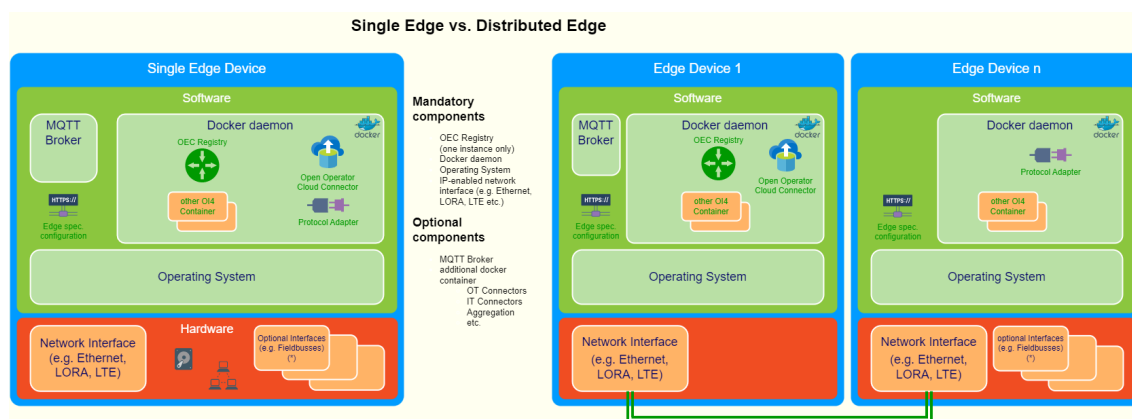


Figure 38: Block diagram of possible Open Edge Computing architectures

To label a device, system or application as “Open Industry 4.0 Alliance Compliant” (see [figure 39](#)), several compliance requirements must be fulfilled.



Figure 39: Badge for OI4 Alliance compliant products

The following subchapters provide an overview about the compliance requirements. For the full list of requirements, the OI4 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 computer 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 OI4 Alliance' perspective, there is a set of requirements that defines the minimum class of edge computers. 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 or similar 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 daemon to run containerized applications must be integrated and available for Open Edge Computing users.
- A MQTT broker must be available on the edge computer or within the Open Edge Computing platform.
- An OEC Registry must be available on the edge computer 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 OI4 Alliance Compliant Application

To achieve seamless application- and vendor-interopability, the OI4 Alliance defines compliance requirements for containerized 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, preferably [multi architecture images](#).
- The container environment-specific requirements need to be fulfilled (6).
- The Message Bus communication requirements must be fulfilled (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 OI4 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 requires 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

A 1 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.

Whenever an application needs to know what other assets (applications and devices) are represented in the Message Bus communication, the OEC Registry can provide the path (`Oi4Identifier`), Master Asset Model (MAM) and Health information.

The OEC Registry's main tasks are:

- Collect information about reachable assets (applications and devices) by listening mainly to MAM and Health resources
- Provide MAM and Health information of all available assets (applications and devices) to others on the Message Bus by offering services to get this information:
 - the topic `Oi4/Registry/<AppId>/Get/MAM` triggers a publication of all master data to the topic `Oi4/Registry/<AppId>/Pub/MAM`.
 - the topic `Oi4/Registry/<AppId>/Get/MAM/<Oi4Identifier>` triggers a publication of a specific master data to the topic `Oi4/Registry/<AppId>/Pub/MAM/<Oi4Identifier>`.
 - the topic `Oi4/Registry/<AppId>/Get/Health` triggers a publication of all Health information to the topic `Oi4/Registry/<AppId>/Pub/Health`.
 - the topic `Oi4/Registry/<AppId>/Get/Health/<Oi4Identifier>` triggers a publication of a specific Health information to the topic `Oi4/Registry/<AppId>/Pub/Health/<Oi4Identifier>`.
- Provide a web front-end, which shows relevant information to its users:
 - List all running Open Industry 4.0 Alliance applications:
 - incl. Master Asset Model (MAM) of the application (10.1.1)
 - incl. last Health information (10.1.2) and availability
 - incl. supported resources (10.1.10) (optional)
 - List all detected devices:
 - incl. MAM of the device (10.1.1)
 - incl. last Health information (10.1.2) and availability

- incl. supported resources (10.1.10) (optional)
- incl. related publisher application (optional)
- Show the audit trail, which was published via `Event` resource to the Message Bus (10.1.9) (optional)

Because of its central position, it might offer additional, but optional, 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.

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 MIT license.*

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.

NOTE *Once an application sends their Birth message, a bunch of `Get` messages from the OEC Registry (community version) are requesting information from the application. The necessary information to request information from the OEC Registry are available through this requests.*

NOTE *With a temporary subscription to `Oi4/Registry/#`, it is possible to detect the `Oi4Identifier` of the OEC Registry. With this information the related topics to request `MAM` and `Health` information can be formed.*

A 2 Predefined DataSetClassIds for Resources of the Alliance

The `DataSetClassId` is an OPC UA 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

Resource (DataSet described in 9.3 ff)	DataSetClassId
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 a04f4f5f-b2cc-40d7-9a6a-0884233ccce7 2b967128-5b7e-4c0f-a6eb-c59af1118ce6 54b8a5c2-812d-442e-8a91-257af7304161 2b4e35a2-437a-4a3d-b007-9b1c27b08127 97439c90-7d10-47b8-8ada-6913a466107d 4292bf9f-b9d0-4de1-a33e-f4736356d4c1 0b3b07af-9cf9-4b40-977f-75b5b02bb0d5 31c6e1c0-88c7-453b-ae9d-85fb6bc50275 cfa96d87-e65c-42f4-9d9e-026797c37f4c 0c959a0f-ec83-43c1-9ccc-20ab20a0e171

Resource (DataSet described in 9.3 ff)	DataSetClassId
	bb725543-9de1-4f22-883f-d32c2de5dea1

Table 14: Predefined DataSetClassIds for resources of the OI4 Alliance

A 3 The dnp-Encoding

The `Oi4Identifier` is a Product Instance URI (hereinafter PIU) which is formed and provided according to the rules of the Open Industry 4.0 Alliance. It is based on information from the nameplate and therefore available for both - greenfield and brownfield.

A [DIN SPEC 91406](#)-compliant PIU does not contain certain special characters, which might be used in product definitions for any brownfield asset. Therefore, the `Oi4Identifier` is build in a way, that masks these special characters, similar to an url-encoding, but with another masking sign.

This so-called dnp-encoding (digital nameplate encoding), can be used for any [DIN SPEC 91406](#) conform PIU and will be published in an appropriate manner.

Define the mask-sign

The set of allowed and prohibited characters is given by the [DIN SPEC 91406](#) (Annex A).

To provide an urlEncode-like encoding, we have to define an algorithm which is easy to understand/implement and (for better acceptance) as similar as possible to the url-encoding:

- use one single character for masking (url: %).
- This character is also used to mask itself (url: % => %25).
- The mask character is followed by a unique definition to represent the masked character (url: %<HEXDIG><HEXDIG>).

The intersection of all reserved characters in DIN SPEC 91406 and masked characters in URL is:

- / ? # [] @ \$ & + , ; =

The intersection of all reserved characters in DIN SPEC 91406 which are not masked in URL is:

- ! ' () *

We define *comma* ($_{\text{ascii}} = 44_{\text{dec}} = 2C_{\text{hex}}$) is the global mask character for dnp-encoding.

NOTE *An comma-encoding mechanism is used to represent a data octet in a component when that octet's corresponding character is outside the allowed set or is being used as a delimiter of, or within, the component. An comma-encoded octet is encoded as a*

character triplet, consisting of the apostrophe character "," followed by the two hexadecimal digits representing that octet's numeric value. For example, ",20" is the comma-encoding for the binary octet "00100000", which in US-ASCII corresponds to the space character (SP).

NOTE The provided dnp-encoding rules are common rules to replace urlEncoding functionality in context of DIN SPEC 91406. It shall not be restricted to the OI4 Alliance or even worst to the `Oi4Identifier`.

How does the dnp-encoding works

dnp-encoding of unreserved characters	
character	Encoded character
a .. z, A .. Z	a .. z, A .. Z
0 .. 9	0 .. 9
- . _ ~	- . _ ~
dnp-encoding of reserved characters	
character	Encoded triplet
#	,23
/	,2F
:	,3A
?	,3F
@	,40
[,5B
]	,5D

dnp-encoding of unreserved characters	
!	,21
\$,24
&	,26
'	,27
(,28
)	,29
*	,2A
+	,2B
,	,2C
;	,3B
=	,3D
dnp-encoding of prohibited printable characters	
character	Encoded tripled
SP	,20
"	,22
%	,25
<	,3C
>	,3E

dnp-encoding of unreserved characters	
\	,5C
^	,5E
`	,60
{	,7B
	,7C
}	,7D

Table 15: dnp-encoding ASCII table

Common usage of dnp-encoding

In general, an URI can exist of various parts such as schema, host, path, query, fragment and others. All these components are divided and marked by reserved characters. This makes it possible to provide machine-readable URIs such as `https://example.com/myproducts?myquery=27#myfragment`. Each reserved character performs a defined function. In case one of the reserved or prohibited characters is part of the path, query or fragment, the character must be masked out to become a machine-readable URI again.

Example:

Let's assume <myquery> points to a product code, which contains spaces (e.g. `Edge Gateway >4GB`)

Furthermore represents <myfragment> a serial number, which contains special characters(e.g. `#20815!2205+1432`).

The resulting DIN SPEC 91406 conform URI would look like this:

```
https://example.com/myproducts?myquery=Edge,20Gateway,20,3E4GB#,2320815,212205,2B1432
```

Using the dnp-encoding for the Oi4Identifier

The basic structure of the `Oi4Identifier` is simple. It consists of `ManufacturerUri` in form of `<manufacturer>.<tld>` and the path consisting of `Model`, `ProductCode` and `SerialNumber`. The components of the path may contain reserved or prohibited characters, so these three components must be dnp-encoded.

We end up with the definition for building the `Oi4Identifier`:

```
ManufacturerUri/dnp-encoded(Model)/dnp-encoded(ProductCode)/dnp-
encoded(SerialNumber)
```

All other rules, provided in chapter 3 are still valid.

Examples:

Manufacturer Uri	Model	ProductCode	SerialNumber	Oi4Identifier
example.com	MyModel	MyProductCode	MySerial	example.com/MyModel/MyProductCode/MySerial
example.com	ABC DEF	123.456/3&8	123456# 33	example.com/ABC,20DEF/123.456,2F3,268/123456,2333
Example.com *	ABC@home	ABC*33<4	20123.4	example.com/ABC,40home/ABC,2A33,3C4/20123.4

*Upper cases in `ManufacturerUri` are prohibited by the DIN SPEC 91406. This rule is covered in chapter 3!

All `Oi4Identifier` are validated with the external [DIN SPEC 91406 validator](#).

Table 16: `Oi4Identifier` examples

A 4 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 OI4 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 OI4 Alliance, each asset has its own Oi4Identifier and 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 OI4 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 (see A3 for details).
dnp-encoding	An url-encoding like mechanism to mask special characters in a product instance uri according to DIN SPEC 91406.
edge computer	An edge computer that fulfills the functional criteria described in this guideline. It is located in the OEC layer. It is alternatively referred to as an "edge device" or "edge gateway". To maintain consistency and prevent potential confusion, this guideline exclusively employs the term "edge computer".
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 OI4 Alliance that provides a critical subset of information about an asset that interacts in an OI4 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 OI4 Alliance. Open Edge Computing mainly deals with the containerized software environment that OI4 Alliance' compliant edge computers 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 platform; The Open Operator Cloud platform is one of the four high level architecture layers of the OI4 Alliance. The Open Operator Cloud platform 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, ...
Open Edge Connectivity	The Open Edge Connectivity layer is one of the four high level architecture layers of the OI4 Alliance Reference Architecture. Open Edge Connectivity mainly provides network bridges from non-Ethernet-based communication to Ethernet-based communication, which can then be further processed by an Open Edge Computing layer (see OEC).
Process Values	In context of the OI4 Alliance, a DataSetMessage called “Process Values” (Oi4Pv, 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 17: Glossary

A 5 Authors

Author	Company
Werner Blumenstock	Siemens AG
Sebastian Czech	Harro Höfliger Verpackungsmaschinen GmbH
Stefan Eggert	M&M Software GmbH
Martin Flöer	Weidmüller GmbH & Co. KG
Dr. Andreas Graf Gatterburg	Hilscher Gesellschaft für Systemautomation mbH
Konrad Heidrich	Hilscher Gesellschaft für Systemautomation mbH
Sebastian Heinze	Christian Bürkert GmbH & Co. KG
Michael Heller	M&M Software GmbH
Vitas Kling	Dunkermotoren GmbH
Matthias Lempertseder	CAPTRON Electronic GmbH
Artur Loorpuu	UReason
Bastian Schmick	ifm electronic gmbh
Matthias Schmidt	ifm electronic gmbh
Peter Sorowka	Cybus GmbH
David Weiß	ifm electronic gmbh
Thomas Weinschenk	Endress+Hauser Digital Solutions (Deutschland) GmbH
Mathis Zeiher	SICK AG

Table 18: Authors (in alphabetical order (last name))

A 6 List of Figures

Figure 1: Open Industry 4.0 Alliance Reference Architecture.....	12
Figure 2: How the Pub/Sub mechanism functions	15
Figure 3: Sequence diagram for asset onboarding	30
Figure 4: Sequence diagram for condition monitoring (Health).....	31
Figure 5: Plain process data sequence diagram	32
Figure 6: Edge data processing sequence diagram.....	32
Figure 7: Store and forward sequence diagram	33
Figure 8: Sequence diagram for the onboarding of a field device	34
Figure 9: Sequence diagram for the condition monitoring (Health) of a field device.....	35
Figure 10: Sequence diagram for the process data ingestion of a field device.....	36
Figure 11: OPC UA PubSub message embedded in the transport protocol.....	76
Figure 12: A simple and valid ua-data JSON	77
Figure 13: A simple and valid ua-metadata JSON.....	78
Figure 14: A simple and valid request MSG JSON	79
Figure 15: A simple and valid response MSG JSON	79
Figure 16: Sequence diagram of the four base interactions on how to use resources over the Message Bus	160
Figure 17: Relationship between topic and payload in standard pub/sub pattern	161
Figure 18: Sequence diagram of the two basic interactions that can be used for the resource MAM.....	162
Figure 19: Sequence diagram of the two basic interactions that can be used for the resource Health	166
Figure 20: Sequence diagram of the three basic interactions that can be used for the resource Config.....	170
Figure 21: Sequence diagram of the two basic interactions that can be used for the resource License	180
Figure 22: Sequence diagram of the two basic interactions that can be used for the resource LicenseText.....	183
Figure 23: Sequence diagram of the two basic interactions that can be used for the resource RtLicense.....	187
Figure 24: Sequence diagram of the three basic interactions that can be used for the resource Data	190
Figure 25: Sequence diagram of the three basic interactions that can be used for the resource Metadata	197
Figure 26: Sequence diagram of the basic interaction that can be used for the resource Event	200
Figure 27: Sequence diagram of the two basic interactions that can be used for the resource Profile	204

- Figure 28: Sequence diagram of the three basic interactions that can be used for the resource PublicationList.....208
- Figure 29: Sequence diagram of the four basic interactions that can be used for the resource SubscriptionList 216
- Figure 30: Sequence diagram of the two basic interactions that can be used for the resource Interfaces 223
- Figure 31: Sequence diagram of the interactions that can be used for the resource ReferenceDesignation 224
- Figure 32: Sequence diagram of the interactions on how to use method calls over the Message Bus 231
- Figure 33: Sequence diagram of the interactions on how to use the method call FileUpload over the Message Bus 232
- Figure 34: Sequence diagram of the interactions on how to use the method call FileDownload over the Message Bus 235
- Figure 35: Sequence diagram of the interactions on how to use the method call FirmwareUpdate over the Message Bus 238
- Figure 36: Sequence diagram of the interactions on how to use the method call Blink over the Message Bus..... 241
- Figure 37: Sequence diagram of the interactions on how to use the method call NewDataSetWriterId over the Message Bus 243
- Figure 38: Block diagram of possible Open Edge Computing architectures 247
- Figure 39: Badge for OI4 Alliance compliant products 247

A 7 List of Tables

- Table 1: IEC 62443 - security level18
- Table 2: DIN SPEC 27070 - Security gateway profile19
- Table 3: Message Bus storage path.....38
- Table 4: Message Bus storage content.....38
- Table 5: OI4 Alliance CA storage path39
- Table 6: OI4 Alliance CA storage content 40
- Table 7: Secret storage..... 40
- Table 8: OI4 Alliance specific application storage path.....42
- Table 9: Advantage/disadvantage - closed network45
- Table 10: Advantages/disadvantages - host network.....46
- Table 11: Advantage/disadvantage - bridge network46
- Table 12: NAMUR NE107 - definition of symbols..... 107
- Table 13: NAMRU NE107 status signals 135
- Table 14: Predefined DataSetClassIds for resources of the OI4 Alliance..... 255
- Table 15: dnp-encoding ASCII table.....259
- Table 16: Oi4Identifier examples.....260
- Table 17: Glossary..... 263
- Table 18: Authors (in alphabetical order (last name)).....264
- Table 19: Document history 278

A 8 History

Revision	Date	Author	Changes/comment
0	2019-07-11	Konrad Heidrich	Initial document
0.1	2019-08-19	Konrad Heidrich	<p>Changed 7.2.1 (Broker address and port)</p> <p>Hint in 7.2.2 for Security Team review</p> <p>Insert 8.1.1 (OI4-element)</p> <p>Changed definitions of 8.1.4 (method element)</p> <p>Started definition of 9 (payload format)</p>
0.2	2019-08-23	Konrad Heidrich	<p>Edited 8.1.4 and 8.1.5 but still not final.</p> <p>Common: some spelling mistakes corrected</p>
0.3	2019-08-26	Konrad Heidrich	<p>Edited 8.1.4 and 8.1.5 (not final).</p> <p>Edit whole chapter 9 (Payload format) (not final)</p> <p>Added and edit chapter 11 (appendix) (not final)</p> <p>Translated whole chapter 8 and 10 to English</p> <p>Common: some spelling mistakes corrected</p> <p>Chapter 11.1 (Docker specific) doesn't belong to this document!</p>
0.4	2019-08-27	Konrad Heidrich, Thomas Weinschenk, Ervin Binkert, Stephan Huber, Christian Scherer	<p>Started 9.3 (Req/Res over OPC UA JSON)</p> <p>Started 8.2 (minimum set of topics)</p> <p>Several changes during T6 meeting</p>
0.5	2019-08-30	Konrad Heidrich	<p>Reworked all chapters as defined in T6 meeting</p> <p>Replaced method data with pub</p> <p>Replaced all resources in plural form with singular form (e.g. devices=> device)</p> <p>Extended and structured chapter 9</p> <p>Renewed chapter 10 and deleted appendix 6.1 (examples)</p>

Revision	Date	Author	Changes/comment
			Added temporary Docker specific Todos (11.1) and Registry specific (11.2) to hold everybody informed
0.6	2019-09-06	Konrad Heidrich, Ervin Binkert, Stephan Huber, Christian Scherer, Manuel Sauer, Matthias Betz, Bastian Schmick, Michael Gernoth	<p>io4-typo fixed to oi4</p> <p>MetaData is not an array of DataSetMetaDataType (9.2.2, 9.1.2, 10.8)</p> <p>DataSetWriterId type conflict in OPCF specification. OI4 assumes it is of type String (changed in 9.2.2, 9.1.2, 10.8).</p> <p>Added CorrelationId in NetworkMessage (9.1.1, 9.2.1, 10.1-10.7, 10.9) and DataSetMetaData (9.1.2, 9.2.2, 10.8).</p> <p>Added payload content to any get-Topic to fulfill CorrelationId needs (10.1 - 10.8).</p> <p>Added the resource profile (8.1.5, 10.11)</p> <p>Added the resource <partial model> (8.1.5, 10.10) to hint to the possibilities.</p> <p>Added General rules (10.12) for collected insights</p> <p>Deleted outdated chapters</p> <p>Added "Questions to OPCF"</p>
0.7	2019-09-06	Andreas Graf Gatterburg, Konrad Heidrich	<p>Relabeling of the document as OI4 Development Guideline, reversioning to 0.7</p> <p>Restructuring of the OI4 Identifier description inside the document from 5.1.3.1 to 2</p> <p>Added Preface (to be completed!)</p> <p>Added description of the Master Asset Model (chapter 3)</p> <p>Added blank chapter for general process definition (chapter 5)</p> <p>Minor reformatting (black section titles, header/footer e.g.)</p> <p>Corrected everywhere to "Edge Computing Platform" and "OI4 Edge Computing"</p> <p>Corrected typo in topic of 8.1.</p> <p>Corrected /device to /mam in chapter 10</p>

Revision	Date	Author	Changes/comment
			<p>Added glossary</p> <p>Added List of Authors</p>
0.8	2019-09-27	Konrad Heidrich	<p>Redefined and described health message in 10.2.</p> <p>Whole document: Marked correlationId as optional, because it is not part of OPC UA and it might not be present for initial messages (if present it contains an empty string).</p> <p>Inserted chapter 11.3 to appendix, which will contain pre-defined DataSetClassIds and its metadata.</p> <p>Reworked chapter 10.9 (handle events)</p>
0.9	2019-10-01	Andreas Graf Gatterburg, Hans-Jürgen Hilscher	<p>Several changes to typos and adding short explanatory sentences throughout the document</p> <p>Erased open points block in the appendix for dissemination outside current committee</p> <p>Removed task leaders from chapters for dissemination outside current committee</p> <p>Added chapter on technical and standards considerations</p> <p>Filled in chapter on the overall process description</p> <p>Removed comments for dissemination outside current committee</p> <p>Complement some diagrams and content from the workshop protocols</p> <p>Expanded Glossary</p> <p>Review</p>
0.10	2019-12-14	Konrad Heidrich	<p>Re-added comments to issues, which are not yet solved</p> <p>Added comments with link to issues when created on trello.</p> <p>Replaced "Edge Computing Platform" with "Open Edge Computing"</p> <p>Changed table for NE107 definitions in 10.2</p>

Revision	Date	Author	Changes/comment
			<p>Added best practice hint on how to map NE107 status to OI4 event levels in 10.9</p> <p>Documented definition differences between IO-Link and OPCF's device status enumeration and how it should be handles in OI4 context (10.9).</p> <p>Added Description object, some corrections and type information to Master Asset Model (4)</p> <p>Added chapter 7.4 to describe general behavior of MQTT on changes.</p> <p>Splitted chapter subRessource Element (8.1.6) for a better overview.</p> <p>Splitted minmum set of topic elementsfor applications (8.2.1) and devices (8.2.2).</p> <p>Started definition of well known DataSetClassIds (11.3).</p> <p>Corrected type information in MAM example (10.1)</p>
0.11	2019-12-15 to 2020-04-03	Konrad Heidrich	<p>Renumbered from chapter 6 and above, because chapter 6 (Introducion to MQTT) was moved to 2.2</p> <p>NE107 status naming was wrong in two cases (9.1.2 and 9.1.9)</p> <p>Insert predefined DataSetClassId GUIDs and started with metadata definition for this resources (10.3)</p> <p>Insert predefined DataSetClassIds to examples in chapter 9.1</p> <p>Adopted GUID description in 8.2.1 and 8.2.4.</p> <p>Added resource publicationList in 7.1.5.1, 9.1.11 and 10.3.9.</p> <p>Add methods req/res in 7.1.4.</p> <p>Describe services in 7.1.5.2 and 7.1.5.3 (in addition to resources)</p> <p>Describe additional MessageType "MSG" in 8.1.3</p> <p>Added several sub-chapters in chapter 8.2 for metadata definition</p> <p>Updated 9.1.2, after ua-metadata information got updated</p>

Revision	Date	Author	Changes/comment
			<p>Added resource subscriptionList in 7.1.5.1, 9.1.12 and 10.3.10.</p> <p>Changed/corrected figures in 8.1 and added them to List of figures</p> <p>Deleted former chapter 9.3 and integrated it into 8.1 and 8.1.3</p> <p>Added chapter Resources (9.1) and moved all former chapters with 9.x to 9.1.x</p> <p>Reworked chapter 7.2</p> <p>Added information for req/res pattern (8.2.16, 8.2.17 and 8.2.18)</p> <p>Added chapter for common services (9.2)</p> <p>Added chapter for specific services (9.3)</p> <p>Changed names of oi4Identifier elements to OPC UA like naming and added information (3.1).</p> <p>Defined assets to be hardware or software (4)</p> <p>deleted hint to SPS'19 in chapter 5</p> <p>Added "Introduction to Edge Computing" (2.1)</p> <p>added "Introduction to OPC UA" (2.3)</p> <p>added "Introduction to Docker" (2.4)</p> <p>Changed/corrected all figures in chapter 5</p> <p>Changed/corrected figure in chapter 2.2.2</p> <p>extended glossary (11)</p> <p>Extended list of authors (12)</p>
0.12	2020-04-15 to 2020-12-16	Konrad Heidrich	<p>Due to moving/adding chapters, a renumbering was necessary</p> <p>Renames Registry to OEC Registry in whole document</p> <p>corrected List of tables and List of figures</p> <p>Type of DataSetWriterId changed to UINT16, as defined in latest OPCF spec</p> <p>Changed oi4Identifier in details, to become DIN SPEC 91406 compliant (3, 3.1 and 3.2)</p>

Revision	Date	Author	Changes/comment
			<p>Corrected typo in figure 4, 5 and 6</p> <p>Chapter 4 described in more detail to be more precise</p> <p>Added information to 5.</p> <p>Added Registry to 5.2.1 its sequence diagram</p> <p>Moved sub chapter Docker and extended/reworked it (6)</p> <p>Added prefix to Docker environment variables (6.2)</p> <p>added max payload size to environment variables (see 6.2)</p> <p>Finalized chapter <i>Docker Image Integrity</i> (6.4)</p> <p>Added information to 7.1, 7.2.2</p> <p>Added Info about “other than publish immediately” to 7.4</p> <p>Added serviceType “ITConnector” (8.1.2) and improved explanations</p> <p>subResource elements for resource event are renamed according to RFC5424 (8.1.6.1)</p> <p>Added DataSetWriterId and additional information to 8.1.7</p> <p>Appended additional key called POI to DataSetMessage (9.1.1, 9.2.3, 10.1ff)</p> <p>Appended additional key called POI to DataSetMetaData (9.1.2, 9.2.2, 10.1.8)</p> <p>Append introduction to chapter 9.2</p> <p>Format of PublisherId concretized, DataSetClassId not mandatory anymore (9.2.1)</p> <p>DataSetClassId set to optional in 9.2.16</p> <p>Added chapter for Alliance' defined objects (9.3) and moved information from 10.1 to it</p> <p>The key healthState got changed to healthScore (9.3.2, 10.1.2)</p> <p>Added implementation of config to 9.3.3 and 10.1.3</p> <p>The key Author got changed to Authors (9.3.4, 10.1.4)</p> <p>Simplified DataSet of license object (9.3.4, 10.1.4)</p>

Revision	Date	Author	Changes/comment
			<p>Extended chapter 9.3.9 (<i>event</i>) for the categories syslog, OPC UA Status Code and NAMUR NE107</p> <p>Added pagination to 9.3.13</p> <p>Added locale to 9.3.14</p> <p>Renamed chapter 10 to <i>Message Bus communication</i></p> <p>Added sequence diagram to 10.1 and all sub-chapters for better understanding</p> <p>Added information on which methods are available for which resource (10.1.x)</p> <p>Added basic information to the payload structure of each resource (10.1.x)</p> <p>Corrections of keys locale and text in 10.1.1</p> <p>Corrected DataSetWriterId to DataSetClassId in explanation (10.1.8).</p> <p>Adopted subResource changes, according to RFC5424, to event description (10.1.9)</p> <p>Added "resource" and concretized the introductory text for publicationList (10.1.11)</p> <p>Changed precision definition, naming (plural) and status definition to active (10.1.11)</p> <p>CREATE_2 and DELETE_4 eliminated in config enumeration of subscriptionList (10.1.12)</p> <p>Reworked chapter 11 (Open Edge Computing platform) and sub chapters</p> <p>Reordered complete Appendix A and Appendix B</p> <p>Added Registry information (A1)</p> <p>Extended glossary (A3)</p> <p>Added additional examples, related to 10.1, to appendix (B1)</p> <p>Moved document history to appendix (A5)</p>
1.0.0	2021-01-11 to 2021-12-16	Konrad Heidrich	<p>Document title changed</p> <p>Spelling and grammar improved throughout the document</p> <p>Renewed several links to external content</p>

Revision	Date	Author	Changes/comment
			<p>Added <i>Conventions</i> chapter (1) to the guideline to explain, how to read this document</p> <p>Actualized whole chapter <i>Overall process description</i> (5)</p> <p>Renamed and reworked whole chapter 6 to <i>Container environment</i></p> <p>Reworked container storage options (6.1 ff)*</p> <p>Deleted mandatory environment variables*</p> <p>Added chapter <i>Container networks</i> (6.4)</p> <p>Changed broker configuration mechanism (7.2.1)*</p> <p>Security measures for Message Bus specified (7.2.2)*</p> <p>Clarified usage of Birth, Close and Will message with an example in 7.3.3, 7.3.4 and 7.3.5</p> <p>Reworked whole chapter 8 to simplify topic schema*</p> <p>Renamed methods req/res to call/reply (8.1.4, 8.1.5, 9.4, 10, 10.2, 10.3)</p> <p>Extended filter options in topic of resource publicationList (8.1.7, 10.1.11)*</p> <p>Extended filter options in topic of resource subscriptionList (8.1.7, 10.1.12)*</p> <p>Extended mandatory resources for devices (8.2.2)</p> <p>Figure about transport protocol structure added to 9</p> <p>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*</p> <p>Added the JSON key “subResource” in 9.1.1, 9.1.2, 9.2.2, 9.2.3 and whole chapter 10.1*</p> <p>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*</p> <p>Reworked ServiceNetworkMessage (9.1.3, 9.2.16)*</p> <p>Clarified usage of DataSetClassId in 9.2.1, 9.2.4 and 9.2.16</p> <p>Added the JSON key “context” to the resource config (9.3.3.1, 9.3.3.2, 10.1.3)*</p>

Revision	Date	Author	Changes/comment
			<p>Added definition of oi4_pv to data resource (9.3.7, 10.1.7)</p> <p>Reworked event (9.3.9, 10.1.9)*</p> <p>Renamed the JSON key “payload” to “details” inside the event resource (9.3.9, 10.1.9)*</p> <p>Added the JSON key “origin” to the event resource (9.3.9, 10.1.9)*</p> <p>Renamed event category “opcSC” to “status” (8.1.6, 8.1.7, 9.3.9, 9.3.9.1 and 10.1.9)*</p> <p>Renamed event category “undef” to “generic” (8.1.6, 8.1.7, 9.3.9, 9.3.9.4 and 10.1.9)*</p> <p>Added sub-chapter 9.3.9.4 for generic events</p> <p>Reworked publicationList (9.3.11, 10.1.11)*</p> <p>Added “subResource” and changed “tag” to “filter”*</p> <p>substituted “active” and “explicit” with “mode”*</p> <p>Changed enumeration for “config”*</p> <p>Added sub chapter 9.4 to explain arguments of Alliance defined methods.</p> <p>Added/defined method “newDataSetWriterId” (9.4.5, 10.2.5)</p> <p>Unified sequence diagrams in chapter 10 to follow the same color schema</p> <p>Added event-mechanism to every “set” or “del” request, done in chapter 10.1*</p> <p>Corrected typo “licText” to “licenseText” in chapter 10.1.5 as defined in 9.3.5</p> <p>Deleted method “findAssets” (former chapter 10.2.1) because this is already covered through other resources.</p> <p>Deleted method “readPV” (former chapter 10.2.2) because this is already covered through other resources.</p> <p>Deleted method “writePV” (former chapter 10.2.3) because this is already covered through other resources.</p> <p>Actualized port information for OEC Registry (A1).</p>

Revision	Date	Author	Changes/comment
			<p>Actualized Glossary (A3)</p> <p>Actualized list of Authors (A4)</p> <p>Actualized appendix B1.1</p> <p>Actualized appendix B1.2</p> <p>*) Breaking changes to guideline v0.12</p>
1.1.0	2022-01-01 to 2022-07-27		<p>Consistent conversion from camelCase to PascalCase in the entire document*</p> <p>Adopted Oi4Identifier in chapter 3.1 to DIN SPEC 91406 compliancy*</p> <p>Added minimal MaxPacketSize in 6.1.1*</p> <p>Concretized 7.2.2</p> <p>Renamed topic element SubResource to Source in 8.1.6 and simplified its usage (10, B1)*</p> <p>Extended Filter for Event from EventLevel to EventCategory/EventLevel in 8.1.7, 10.1.9 and B1.1.9*</p> <p>Changed predefined Data Tag from oi4_pv to Oi4Pv (8.1.7, 10.1.7 and B1.1.7)*</p> <p>Added ReferenceDesignation as mandatory to 8.2.1*</p> <p>Renamed SubResource to Source in DataSetMessage (9.2.3, 10.1 ff)*</p> <p>Added requirement information to 9.2.15</p> <p>Renamed keys of predefined DataSet for Oi4Pv (9.3.7, 10.1.7 and B1.1.7)*</p> <p>Deleted Origin in Event object (9.3.9 ff, 10.1 ff)*</p> <p>Changed Resource[] to Resources[] in Profile (9.3.10, 10.1.10, B1.1.10)*</p> <p>The key Oi4Identifier is not needed anymore in PublicationList (9.3.11, 10.1.11)*</p> <p>Pagination and PaginationRequest are different objects now (9.3.15, 9.3.15.1, 9.3.15.2)*</p> <p>The Pagination object is extended with a PaginationId (9.3.15.2)*</p> <p>Extended description of OEC Registry (A1)</p>

Revision	Date	Author	Changes/comment
			<p>Inserted A3 to explain dnp-encoding</p> <p>Extended glossary (A4)</p> <p>Minor editorial changes such as typos or external links</p> <p>*) Breaking changes to guideline v1.0</p>
1.1.1	2023-01-01 to 2023-07-31	Konrad Heidrich Lucas Wolf	<p>Added SLO in table 1 as clarification</p> <p>Added “Access:” and “Requirement:” to type definition in 3.1</p> <p>deleted outdated hints to “Device Driver” in 5.1.3</p> <p>“Filter” in topic and payload corrected to <topic encoded string>, was <url-encoded> before (s. 8.1.7).</p> <p>Hirarchy structure in 9.3.9.3 corrected.</p> <p><Filter> in sequence diagrams replaced by context-sensitive name (s. 10.1 ff).</p> <p><Source> in sequence diagrams replaced by <Oi4Identifier> (s. 10.1 ff).</p> <p>Elimination of spelling errors and improvements in expression for better readability.</p>

Table 19: Document history

Appendix B

B 1 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`

B 1.1 Resources

All following examples are related to chapter [10.1](#).

B 1.1.1 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 55,
      "Source": "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": 55,
      "Source": "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 objects 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/provider.com/FieldDataService/FDS-001/1200-
0345/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": "000000000000-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": 55,
      "Source": "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"
        }
      }
    }
  ]
}
```


B 1.1.2 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "Source": "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,
      "Source": "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 objects 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/provider.com/FieldDataService/FDS-001/1200-0345/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": "000000000000-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",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Health": "NORMAL_0",
        "HealthScore": 0
      }
    }
  ]
}
```

Will message of an application

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/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": "000000000000-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",
  "Messages": [
    {
      "DataSetWriterId": 3456,
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Health": "FAILURE_1",
        "HealthScore": 0
      }
    }
  ]
}
```

B 1.1.3 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>",
      "Source": "<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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "Filter": "Network%20settings",
      "Source": "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",
      "Source": "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 objects 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",
      "Source": "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 objects 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 SubscriptionList 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",
      "Source": "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 (next site):


```

{
  "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",
      "Source": "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",
      "Source": "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",
      "Source": "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/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good
```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Statu/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 42,
      "Filter": "Overall%20device%20settings",
      "Source": "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 (next site):

```
{
  "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",
      "Source": "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 (next site):


```

{
  "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",
      "Source": "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 (next site):

```
{
  "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",
      "Source": "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/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good
```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}

```

B 1.1.4 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 357,
      "Filter": "MIT",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 21,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Components": [
          {
            "Component": "MQTTCLI",
            "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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 357,
      "Filter": "MIT",
      "Source": "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",
      "Source": "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 objects 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",
      "Source": "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",
      "Source": "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 objects are existing, a pagination object would point  
that out  
  //and further NetworkMessages with License objects would be published under  
the same topic  
}  
]  
}
```

B 1.1.5 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": []
}

```

```

{
  "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",
      "Source": "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:\n
The 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 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.*

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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 489,
      "Filter": "MIT",
      "Source": "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:\n\nThe above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.\n\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",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "LicenseText": "Licensed under the Apache License, Version 2.0 (the \"License\"); you may not use this file except in compliance with the License.\n\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."
```

```
  }\n},\n{\n  //additional LicenseText objects would be listed here if existing\n},\n{\n  //in case too many LicenseText objects are existing, a pagination object would  
point that out\n  //and further NetworkMessages with LicenseText objects would be published  
under the same topic\n}\n]\n}
```

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",
      "Source": "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:\n
The above copyright notice and this
permission notice shall be included in all copies or substantial portions of the
Software.\n
THE 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",
      "Source": "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.\n
You may obtain a copy of the License at\n
  http://www.apache.org/licenses/LICENSE-2.0\n
Unless 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 objects are existing, a pagination object would  
    point that out  
    //and further NetworkMessages with LicenseText objects would be published  
    under the same topic  
  }  
]  
}
```

B 1.1.6 RtLicense

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

B 1.1.7 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/oe
```

```

{
  "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 OI4 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "Filter": "oee",
      "Source": "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",
      "Source": "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 objects 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": "oe",
      "Source": "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 objects 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/oe

```

NOTE *Not every DataSet is writable over the Message Bus. Check SubscriptionList 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",
      "Source": "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/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good

```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}

```

B 1.1.8 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/oe

```

```
{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES,2fOnPrem/Inst07322",
  "MessageType": "ua-metadata",
  "PublisherId": "ITConnector/consumer.com/MEService/MES,2fOnPrem/Inst07322",
  "DataSetWriterId": 753,
  "Filter": "oee",
  "Source": "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/oee
```

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",
  "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
  "CorrelationId": "1639693510000-ITConnector/consumer.com/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/oee

```

```

{
  "MessageId": "1639693510000-ITConnector/consumer.com/MES,2fOnPrem/Inst07322",
  "MessageType": "ua-metadata",
  "PublisherId": "ITConnector/consumer.com/MES,2fOnPrem/Inst07322",
  "DataSetWriterId": 753,
  "Filter": "oee",
  "Source": "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/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good

```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 38,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}
```

B 1.1.9 Event

NOTE Only `DataSetMessages` of type `Event` with same `EventCategory` and `EventFilter` can be combined in a single `NetworkMessage`.

Pub Event of `EventCategory Status (OPC UA Statuscode)`

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/Event/provider.com/FieldDataService/FDS-001/1200-0345/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": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 14,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}
```

Pub Event of EventCategory Syslog

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/Event/provider.com/FieldDataService/FDS-001/1200-0345/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 EventCategory 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": "/Syslog/Notice",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 15,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": <RPI>,
        "Category": "CAT_SYSLOG_0"
        "Details": {
          "MSG": "<MSG>",
          "HEADER": "<HEADER>"
        }
      }
    }
  ]
}

```

Pub Event of EventCategory Ne107 (NAMUR NE107)

```

Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/Event/provider.com/FieldDataService/FDS-001/1200-0345/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": "Ne107/MaintenanceRequired",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 16,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 4,
        "Description": "Maintenance Required",
        "Category": "CAT_NE107_2"
        "Details": {
          "DiagnosticCode": "F-238",
          "Location": ""
        }
      }
    }
  ]
}
```

Pub Event of EventCategory Generic

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/Event/provider.com/FieldDataService/FDS-001/1200-0345/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": "Generic/High",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 17,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": <application defined number>,
        "Description": "<application defined description>",
        "Category": "CAT_GENERIC_99"
        "Details": {
          <application defined object>
        }
      }
    }
  ]
}

```

B 1.1.10 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

```

{
  "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,
      "Filter": "",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 42,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Resources": [

```

```

    "Mam",
    "Health",
    "License",
    "LicenseText",
    "RtLicense",
    "Profile",
    "PublicationList",
    "SubscriptionList"
  ]
}
}
]
}

```

```
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.*

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,
      "Filter": "",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resources": [
          "Mam",
          "Health",
          "License",
          "LicenseText",
          "RtLicense",
          "Profile",
          "PublicationList",
          "SubscriptionList"
        ]
      }
    },
    {
      //additional Profile objects would be listed here if existing
    },
    {
      //in case too many Profile objects are existing, a pagination object would point
      that out
      //and further NetworkMessages with Profile objects would be published under the
      same topic
    }
  ]
}

```

B 1.1.11 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/[<Source>[<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/oe
```

```
{
  "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/oe
```

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",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "Data",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
        "Filter": "oee",
        "DataSetWriterId": 93,
        "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",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "MAM",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 55,
        "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "Config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "Filter": "Health",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "Health",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 3456,
        "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "Interval": 0,
        "Config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "Filter": "Profile",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "Profile",

```

```

    "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
    "DataSetWriterId": 74,
    "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
    "Interval": 0,
    "Config": "NONE_0"
  }
},
{
  "DataSetWriterId": 161,
  "Filter": "Event",
  "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
  "Payload": {
    "Resource": "Event",
    "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
    "Filter": "Syslog/Notice",
    "DataSetWriterId": 625,
    "Mode": "APPLICATION_2",
    "Interval": 0,
    "Config": "NONE_0"
  }
},
{
  "DataSetWriterId": 161,
  "Filter": "Data",
  "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
  "Payload": {
    "Resource": "Data",
    "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
    "Filter": "oee",
    "DataSetWriterId": 93,
    "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
    "Config": "MODE_AND_INTERVAL_3"
  }
}
},
{
  //additional PublicationList objects would be listed here if existing
},
{

```

```
//in case too many PublicationList objects 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",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "MAM",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 55,
        "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "Config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "Filter": "Health",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "Health",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
        "DataSetWriterId": 3456,
        "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
        "Interval": 0,
        "Config": "NONE_0"
      }
    },
    {
      "DataSetWriterId": 161,
      "Filter": "Profile",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "Profile",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",

```

```

    "DataSetWriterId": 74,
    "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
    "Config": "NONE_0"
  }
},
{
  "DataSetWriterId": 161,
  "Filter": "Event",
  "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
  "Payload": {
    "Resource": "Event",
    "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
    "Filter": "notice",
    "DataSetWriterId": 625,
    "Mode": "APPLICATION_2",
    "Config": "NONE_0"
  }
},
{
  "DataSetWriterId": 161,
  "Filter": "Data",
  "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
  "Payload": {
    "Resource": "Data",
    "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
    "Filter": "oee",
    "DataSetWriterId": 93,
    "Mode": "APPLICATION_SUBRESOURCE_FILTER_8",
    "Config": "MODE_AND_INTERVAL_3"
  }
},
{
  //additional PublicationList objects would be listed here if existing
},
{
  //in case too many PublicationList objects 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",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Resource": "Data",
        "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
        "Filter": "Network%20settings",
        "DataSetWriterId": 93,
        "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/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good
```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 39,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}

```

Excursion on how to use precision:

DataSet (example)	Precisions (full array for DataSet example)	Precisions (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	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	We can reduce the number of Precision objects, when we only list the values, which shall use other than default (on change) precision:

DataSet (example)	Precisions (full array for DataSet example)	Precisions (shorted array for DataSet example)
values inside the DataSet are quite "floating":	functionality - "temperature" and "speed":	
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:
<pre>"Payload": { "temperature": 37.2, "speed": 1500, "torque": 44, "direction": "left" }</pre>	<pre>"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 }</pre>	<pre>"Precisions": { "temperature": 0.5, //next pub would be on <=36.7 or >=37.7 "speed": 10 //next pub would be on <=1490 or >=1510 }</pre>
	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).

B 1.1.12 SubscriptionList

NOTE Depending on which resource a searched SubscriptionList 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>/SubscriptionList/[<Source>[/<ResourceType>[/<Tag>]]]
```

NOTE For a SubscriptionList entry belonging to the resource Data, the Filter is a combination of ResourceType and Tag.

NOTE For a `SubscriptionList` entry belonging to the resource `MAM`, the `Filter` consists only of `ResourceType`.

Get explicit `SubscriptionList` entry

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Get/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-0345/Data/oe
```

```
{
  "MessageId": "1639693510000-ITConnector/consumer.com/MEService/MES,2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId": "ITConnector/consumer.com/MEService/MES,2fOnPrem/Inst07322",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "CorrelationId": "",
  "Messages": []
}
```

Get all `SubscriptionList` entries of an explicit asset

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Get/SubscriptionList/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": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "CorrelationId": "",
  "Messages": []
}
```

Get full `SubscriptionList` of all assets related to this application

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Get/SubscriptionList
```

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": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "Messages": []
}
```

Pub an explicit SubscriptionList entry

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/Pub/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-
0345/Data/oe
```

NOTE *Let's assume, this publication was requested by previous published ...Get/SubscriptionList/... - 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": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "CorrelationId": "1639693510000-ITConnector/consumer.com/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 227,
      "Filter": "Data/oe",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "TopicPath": "Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-0345/Data/oe",
        "Interval": 0,
        "Config": "CONF_1"
      }
    }
  ]
}

```

Pub all SubscriptionList entries of an explicit asset

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/SubscriptionList/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": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "CorrelationId": "",
  "Messages": [
    {
      "DataSetWriterId": 227,
      "Filter": "Data/oe",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "TopicPath": "Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-0345/Data/oe",
        "Interval": 0,
        "Config": "CONF_1"
      }
    },
    {
      //additional config objects would be listed here if existing
    },
    {
      //in case too many config objects are existing, a pagination object would point that out
      //and further NetworkMessages with config objects would be published under the same topic
    }
  ]
}

```

Pub full SubscriptionList of all assets related to this application

Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/SubscriptionList

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": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "Messages": [
    {
      "DataSetWriterId": 227,
      "Filter": "Data/oe",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "TopicPath": "Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-0345/Data/oe",
        "Interval": 0,
        "Config": "CONF_1"
      }
    },
    {
      //additional config objects would be listed here if existing
    },
    {
      //in case too many config objects are existing, a pagination object would point that out
      //and further NetworkMessages with config objects would be published under the same topic
    }
  ]
}Del

```

Set an explicit SubscriptionList entry

```

Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Set/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-0345/Data/oe

```

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 `SubscriptionList` 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": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "Messages": [
    {
      "DataSetWriterId": 227,
      "Filter": "Data/oe",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "TopicPath": "Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/Pub/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-
0345/Data/oe",
        "Interval": 1000,
        "Config": "CONF_1"
      }
    }
  ]
}
```

Status event follows a set request

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/Pub/Event/provider.com,2FFieldDataService,2FFDS-001,2F1200-
0345/Status/Good
```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}

```

Del an explicit SubscriptionList entry

```

Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Del/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-0345/Data/oe

```

```

{
  "MessageId": "1639693510000-
ITConnector/consumer.com/MEService/MES,2fOnPrem/Inst07322",
  "MessageType": "ua-data",
  "PublisherId": "ITConnector/consumer.com/MEService/MES,2fOnPrem/Inst07322",
  "DataSetClassId": "e5d68c47-c276-4929-8ab9-4c1090cac785",
  "Messages": [
    {
      "DataSetWriterId": 227,
      "Filter": "Data/oe",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "TopicPath": "Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/Pub/SubscriptionList/provider.com/FieldDataService/FDS-001/1200-
0345/Data/oe",
        "Interval": 1000,
        "Config": "CONF_1"
      }
    }
  ]
}

```

Status event follows a delete request

```

Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/Pub/Event/provider.com,2FFieldDataService,2FFDS-001,2F1200-
0345/Status/Good

```

NOTE Every delete request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related delete request. The `CorrelationId` of the event equals to the `MessageId` of the initial delete 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}
```

B 1.1.13 Interface

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

B 1.1.14 ReferenceDesignation

Get ReferenceDesignation of an explicit asset

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Get/ReferenceDesignation/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": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "",
  "Messages": []
}
```

Get ReferenceDesignation of all assets related to this application

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Get/ReferenceDesignation
```

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": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "Messages": []
}
```

Pub ReferenceDesignation of an explicit asset

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-
0345/Pub/ReferenceDesignation/provider.com/FieldDataService/FDS-001/1200-
0345
```

NOTE *Let's assume, this publication was requested by previous published `...Get/ReferenceDesignation/...` - the `CorrelationId` points to the `MessageId`, which requested this publication.*

NOTE *The `Filter` is either empty or omitted, because `ReferenceDesignation` 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": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "1639693510000-ITConnector/consumer.com/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "Filter": "",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 4,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Location": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Parent": {
            "Value": "<designation Value>",
            "Local": "<Local designation Value>",
            "Oi4Identifier": "<Oi4Identifier>"
          }
        }
      },
      "Function": {
        "Value": "<designation Value>",
        "Local": "<Local designation Value>",
        "Parent": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      },
      "Product": {
        "Value": "<designation Value>",
        "Local": "<Local designation Value>",
        "Parent": {
          "Value": "<designation Value>",

```



```
"Local": "<Local designation Value>",  
"Oi4Identifier": "<Oi4Identifier>"  
  }  
}  
}  
}  
]  
}
```

Pub ReferenceDesignation of all assets related to this application

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-  
0345/Pub/ReferenceDesignation
```

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": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Location": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Parent": {
            "Value": "<designation Value>",
            "Local": "<Local designation Value>",
            "Oi4Identifier": "<Oi4Identifier>"
          }
        }
      },
      "Function": {
        "Value": "<designation Value>",
        "Local": "<Local designation Value>",
        "Parent": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      },
      "Product": {
        "Value": "<designation Value>",
        "Local": "<Local designation Value>",
        "Parent": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      }
    }
  ]
}

```

```
    }  
  },  
  {  
    //additional mam objects would be listed here if existing  
  },  
  {  
    //in case too many mam objects are existing, a pagination object would point that  
out  
    //and further NetworkMessages with mam objects would be published under the  
same topic  
  }  
]  
}
```

Set ReferenceDesignation of an explicit asset

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-  
0345/Set/ReferenceDesignation/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": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "Messages": [
    {
      "DataSetWriterId": 93,
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "Payload": {
        "Location": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Parent": {
            "Value": "<designation Value>",
            "Local": "<Local designation Value>",
            "Oi4Identifier": "<Oi4Identifier>"
          }
        }
      },
      "Function": {
        "Value": "<designation Value>",
        "Local": "<Local designation Value>",
        "Parent": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      },
      "Product": {
        "Value": "<designation Value>",
        "Local": "<Local designation Value>",
        "Parent": {
          "Value": "<designation Value>",
          "Local": "<Local designation Value>",
          "Oi4Identifier": "<Oi4Identifier>"
        }
      }
    }
  ]
}

```

```
}  
]  
}
```

Status Event follows a set request

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-  
0345/Pub/Event/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good
```

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. The `CorrelationId` of the event equals to the `MessageId` of the initial 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}

```

Del ReferenceDesignation of an explicit asset

```

Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Del/ReferenceDesignation/provider.com/FieldDataService/FDS-001/1200-0345

```

```
{
  "MessageId": "1639693510094-OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "MessageType": "ua-data",
  "PublisherId": "OTConnector/provider.com/FieldDataService/FDS-001/1200-0345",
  "DataSetClassId": "27a75019-164a-496d-a38b-90e8a55c2cfa",
  "CorrelationId": "1639693510000-ITConnector/consumer.com/MESService/MES,2fOnPrem/Inst07322",
  "Messages": []
}
```

Status Event follows a delete request

```
Oi4/OTConnector/provider.com/FieldDataService/FDS-001/1200-0345/Pub/Event/provider.com/FieldDataService/FDS-001/1200-0345/Status/Good
```

NOTE Every delete request over the Message Bus, somebody is subscribed to, triggers a status event. The event contains status information about the execution of the related delete request. The `CorrelationId` of the event equals to the `MessageId` of the initial delete 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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 623,
      "Filter": "Status/Good",
      "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
      "SequenceNumber": 37,
      "Timestamp": "2021-12-16T22:25:10.094Z",
      "Payload": {
        "Number": 0,
        "Description": "The operation succeeded.",
        "Category": "CAT_STATUS_1"
        "Details": {
          "SymbolicId": "Good"
        }
      }
    }
  ]
}
```

B 1.1.15 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,
      "Filter": "PaginationRequest",
      "Source": "consumer.com/MEService/MES,2fOnPrem/Inst07322",
      "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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 55,
      "Source": "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,

```

```
"Source": "groee.mca/DeviceModelB/ModelA/AC33.447/20443",
"Payload": {
  "Manufacturer": {
    "Locale": "en-US",
    "Text": "GROE"
  },
  "ManufacturerUri": "https://groee.mca",
  "Model": {
    "Locale": "en-US",
    "Text": "DeviceModelB"
  },
  "ProductCode": "AC33.447",
  "HardwareRevision": "1.1",
  "SoftwareRevision": "2.4.1",
  "DeviceRevision": "4",
  "DeviceManual": "",
  "DeviceClass": "",
  "SerialNumber": "20443",
  "ProductInstanceUri": "groee.mca/DeviceModelB/AC33.447/20443",
  "RevisionCounter": 1,
  "Description": {
    "Locale": "en-US",
    "Text": "Device Model B measurement unit"
  }
}
},
{
  "DataSetWriterId": 1,
  "Filter": "Pagination",
  "Source": "provider.com/FieldDataService/FDS-001/1200-0345",
  "Payload": {
    "TotalCount": 2,
    "PerPage": 25,
    "Page": 1,
    "HasNext": false,
    "PaginationId": "1639693510094-
OTConnector/provider.com/FieldDataService/FDS-001/1200-0345"
  }
}
```

```

]
}

```

B 1.1.16 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,
      "Filter": "Locale",
      "Source": "consumer.com/MEService/MES,2fOnPrem/Inst07322",
      "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/MESService/MES,2fOnPrem/Inst07322",
  "Messages": [
    {
      "DataSetWriterId": 55,
      "Source": "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,

```

```
"Source": "groee.mca/DeviceModelB/ModelA/AC33.447/20443",
"Payload": {
  "Manufacturer": {
    "Locale": "en-US",
    "Text": "GROE"
  },
  "ManufacturerUri": "https://groee.mca",
  "Model": {
    "Locale": "en-US",
    "Text": "DeviceModelB"
  },
  "ProductCode": "AC33.447",
  "HardwareRevision": "1.1",
  "SoftwareRevision": "2.4.1",
  "DeviceRevision": "4",
  "DeviceManual": "",
  "DeviceClass": "",
  "SerialNumber": "20443",
  "ProductInstanceUri": "groee.mca/DeviceModelB/AC33.447/20443",
  "RevisionCounter": 1,
  "Description": {
    "Locale": "en-US",
    "Text": "Device Model B measurement unit"
  }
}
}
```

B 1.2 Common Services

All following examples are related to Chapter [10.2](#).

B 1.2.1 FileUpload

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

B 1.2.2 FileDownload

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

B 1.2.3 FirmwareUpdate

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

B 1.2.4 Blink

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

B 1.2.5 NewDataSetWriterId

The IT connector is able to provide information to generate the `ReferenceDesignation` of the OT connector. To do so, the IT connector needs a valid `DataSetWriterId` from the OT connector, 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/MESService/MES%2fOnPrem/Inst07322",
  "Message": {
    "Results": [
      {
        "StatusCode": 0,
        "InputArgumentResults": [0],
        "OutputArguments": [
          {
            "DataSetWriterId": 57211,
            "Ttl": 10
          }
        ]
      }
    ]
  }
}
```

