OPC UA Information Model Kit for manufacturing use cases

SiOME / Version 3.0

https://support.industry.siemens.com/cs/ww/en/view/109814835

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under https://www.siemens.com/cert.

Table of contents

1 Introduction

1.1 Purpose of the document

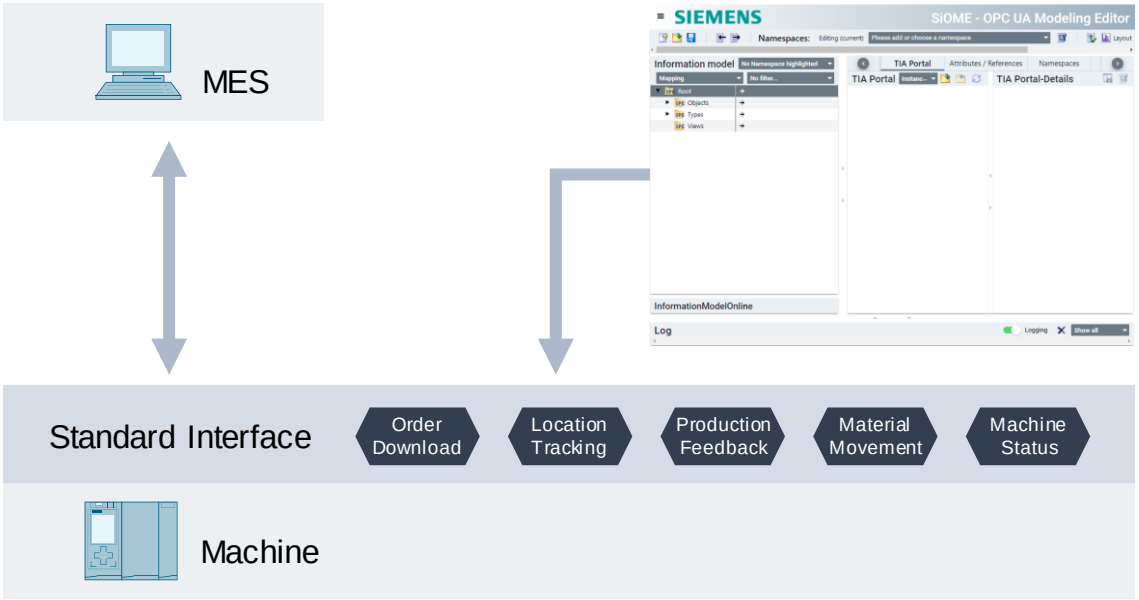
Interoperability of production assets is a key factor for modern production environments. It is crucial to exchange information among the individual equipment executing the actual manufacturing processes on the shop floor with the supervisory control level in order to execute the manufacturing processes efficiently. Furthermore, the integration of several systems is a constant necessity to maintain flexibility of a production system.

Standardized communication concepts such as OPC UA supply the base for successfully integrating the equipment control level with the higher levels of factory automation, such as the Manufacturing Execution System (MES) Level. Fostering the open standard helps in integrating machinery of different suppliers as given in most of the industries. Although, industries differ strongly when it comes to details on processes and equipment, there is an overlap regarding manufacturing use cases.

With this OPC UA information model, a concept for standardizing the interface for production machine is introduced. It is crucial to mention, that this model focuses on the topic of the integration of IT systems and not on M2M communication.

The presented document, therefore, shall serve as a starting point for implementing production systems with an adequate modern automation information architecture, as well as a guidance for OEMs to design their equipment to be suitable to be easily integrated in a respective environment.

Figure 1-1: Using the Siemens Informationmodel Kit to create a standardized machine interface



1.2 Background

1.2.1 Standardization

The idea of standardization is of course the idea of "plug-and-play". It is supposed to be easy to just plug in a machine with a standard interface directly to any integration layer and to be able to access data out-of-the-box. To achieve that, there are two aspects of interfaces that need to be standardized:

- The **technical interface** is the communication mechanism, the protocol, or the technology how to transfer data. Consortia and global players over the world decided to focus on OPC UA as the standard to implement the technical interface. With its advantages like the different communication functionalities like subscriptions, methods, PubSub etc., an implementation of security and the possibility to specify object-oriented information models, OPC UA has become the first choice of standardized communication.
- The **functional interface** is the logic behind the data that is transmitted. It contains the data structure and names of the sent or received information and the meaning that it has to both, the sender and receiver. In OPC UA it is possible to define these data structure within an Information model in the format of a standard OPC UA Nodeset. This document shows how to do that.

Often a standardized interface is reduced to the technical interface, missing the functional part. This is like connecting a printer to your PC by just assembling a USB plug to the printer, without defining nor implementing APIs or drivers to communicate. You might be able to plug in the printer, but you cannot print your documents, neither in black nor in color.

Next to the reduction of a standardized interface to only the technical interface, another problem is, that not all of the use cases, aimed to reach, are sufficiently specified and hence implemented.

1.2.2 IT/OT Integration

When it comes to IT/OT integration, people are talking about different things. To avoid misunderstandings, we need to distinguish between at least two separate fields-of-use:

Data management and Monitoring

This field contains the integration of machine or material data to IT systems to save and provide data. Popular use cases are:

- KPI / OEE calculation
- Dashboards
- Monitoring
- AI & Analytics
- Energy data & root cause analysis
- Historian

The data used for these consist of counters, states, parameters etc. that are continuously changing over time and thus independent from other data. If a machine can and should be controlled by just a tag (usually valid in integrations of SCADA systems or for M2M communication) this can be a use case here as well. The possibly best-known Information Model in this field is OMAC PackML, but also nearly every other Companion Specification. It provides a data model for OEE calculation, root cause analysis, for SCADA and line/M2M integration and the full blown state model that is, regardless of its advantages, slightly too extensive for most of the IT/OT integrations. As there are currently many state models and objects defined in different companion specification, the Siemens Information Model Kit tries to harmonize them and provide a best-practice approach from experience.

Digital Factory Integration

The second field-of-use cases of IT/OT integration is focused on a strong interaction between IT and OT systems. Particularly MOM systems, but also ERP or other systems, are integrated to implement use cases like:

- Order Download
- Recipe/Parameter Download
- Production Feedback
- Process Parameter Upload
- Tracking & Tracing (Movements/Consumptions)
- MachineInterlocking
- …and more

Where the Data Management use cases are thought of just observing the machine data from IT systems, here it is usually necessary to interact with the machine and communicate event-based to send contextualized commands or notifications. The data to be sent are not only single tags to observe, but whole messages out of i.e. a source and a destination location, that need to arrive completely at a point of time.

Using OPC UA as the standard communication technology, the straightforward features to communicate event-based and consistently are OPC UA Methods and OPC UA Alarms & Conditions. As not all systems are implementing the full specification of OPC UA, sometimes it is necessary to bypass this with the usage of handshakes using what is implemented. For further information on the different handshake mechanisms with or without methods, see 2.2 and 2.3 for details and https://support.industry.siemens.com/cs/ww/en/view/109795979 for further information.

Implementing Digital Factory Integration, it is hard to define one standard interface that fits all machines and all customers. Events that should be sent are not only very individual to the process, but they also need to be specifically implemented on the machine and the IT system to be handled. It would be too costly (and too confusing) to implement all of the possible use cases per default on each machine. For this, in the best case the OEM can specify his own standard interface for at least parts of his production.

To create a standardized information model for its production, the OEM needs to meet with its machine builders and IT system administrators to discuss about the information needed and to compare it with the already accessible data provided. This leads to the situation that professionals from two different worlds, the real-time tag-based OT world, and the distributed service-based IT world need to communicate and find common sense to agree to an OPC UA server interface which connected systems are able to handle. To ease the communication and save time at creation, the tool SiOME (Siemens OPC UA Modelling Editor) was created to support specifying a standard nodeset that can be directly imported to TIA Portal or of course every other OPC UA Server system that can import standard nodesets. The tool is published and can be downloaded here: https://support.industry.siemens.com/cs/ww/en/view/109755133.

1.3 Components used

This application example has been created with the following software components:

Table 1-1

Component	Note
SiOME 2.5.0	https://support.industry.siemens.com/cs/ww/en/view/109755133

This application example consists of the following components:

Table 1-2

Component	Note
Base information model kit NodeSet	BaseInformationmodelKit_3.0.xml
Base information model example NodeSet	BaseInformationmodelExample_3.0.xml
Information model kit NodeSet	InformationmodelKit_3.0.xml
Information model kit example NodeSet using methods	InformationmodelExampleMethodBased_3.0.xml
Information model kit example NodeSet using tags	InformationmodelExampleTagBased_3.0.xml
Information model for Opcenter Execution Discrete NodeSet	InformationmodelDiscrete_3.2.xml
Information model instances for Opcenter Execution Discrete NodeSet	InformationmodelDiscreteInstances_3.2.xml
Information model for Opcenter Execution Process NodeSet	InformationmodelProcess_3.0.xml
Information model instances for Opcenter Execution Process NodeSet	InformationmodelProcessInstances_3.0.xml
Alarming Extension NodeSet	InformationmodelKitAlarming_3.0.xml
Alarming Extension example NodeSet	InformationmodelExampleAlarms_3.0.xml
Energy Monitoring Extension NodeSet	InformationmodelKitEnergy_3.0.xml
Energy Monitoring Extension example NodeSet	InformationmodelExampleEnergy_3.0.xml

2 Concept

2.1 Communication

The OPC UA protocol is a platform independent and service-oriented communication protocol. It defines various features to implement a variety of integration uses cases for the bidirectional information flow between automation level (controls) with the IT world. For this document we must respect the fact that today many OPC UA features are not yet common standard on all IT and OT levels. For example, some IT applications might not yet offer integrated OPC UA services like OPC UA alarms and condition or methods.

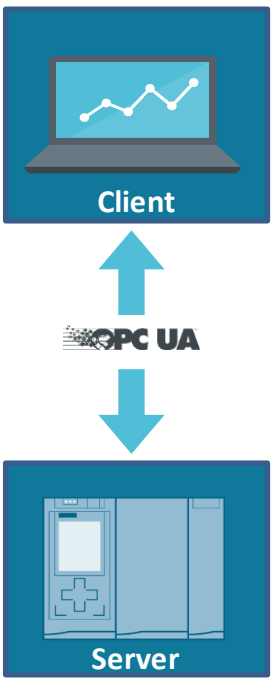
For this reason, this document recommends a communication concept reducing the number of OPC UA requirements to a minimum for the automation level, respectively the PLCs. The communication exchange assumes that the IT layer does support the OPC UA Client services (registered) read / write and subscription.

The focus of the document is to provide an OPC UA nodeset for different integration scenarios in the scope of a production. For a consistent bidirectional communication as well as the IT level and shopfloor level must provide the following OPC UA functionalities:

- (Registered) Read / Write
- Subscription

The main architecture of the communication is described in Figure 2-1. The machine is considered as an OPC UA server and the higher-level system as an OPC UA client.

Figure 2-1: OPC UA client and server architecture



Within the production process, the machine must be able to report information to the IT system for several scenarios, e.g. Tracking and Tracing. Two scenarios need to be differentiated:

1. Continuous data exchange, e.g. process data based on subscriptions.
2. Message-based data exchange based on a handshake mechanism.

For the second scenario, it is necessary to provide a bidirectional, consistent and reliable data exchange. Therefore, two different communication procedures are defined.

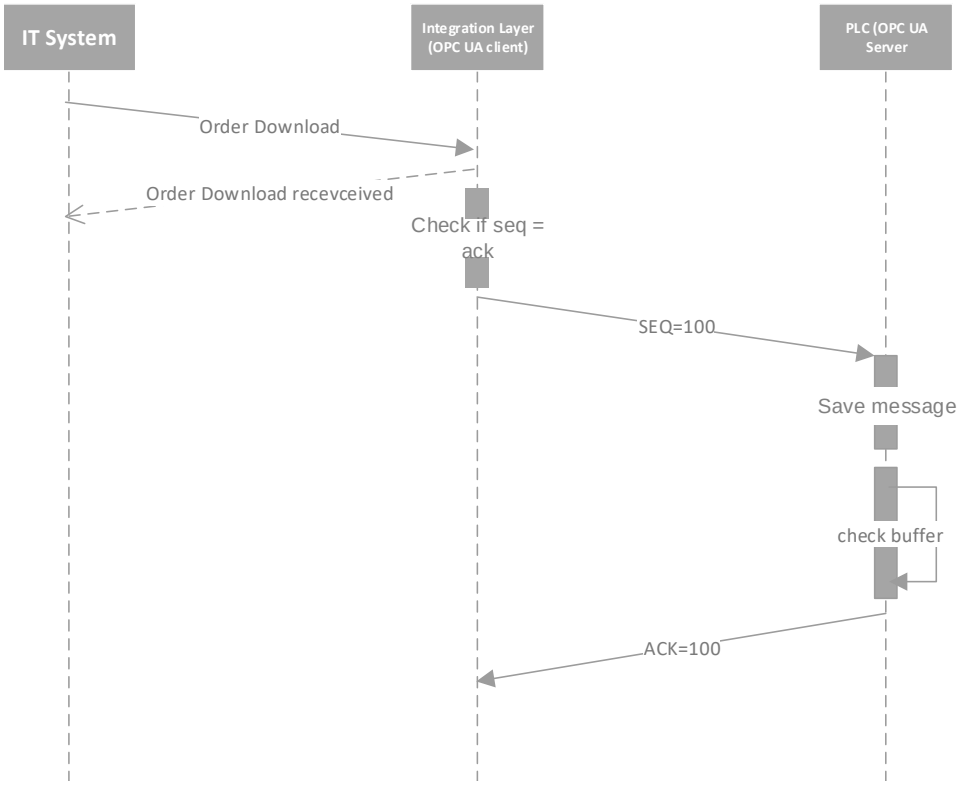
2.2 Handshakes in tag-based integration

To exchange data via an OPC UA handshake, the following mechanism, as described below, is defined.

OPC UA client sends data to OPC server on the PLC.

The following scenario assumes that the integration layer does persistently buffer the data until acknowledged by the PLC:

Figure 2-2: Inbound communication process

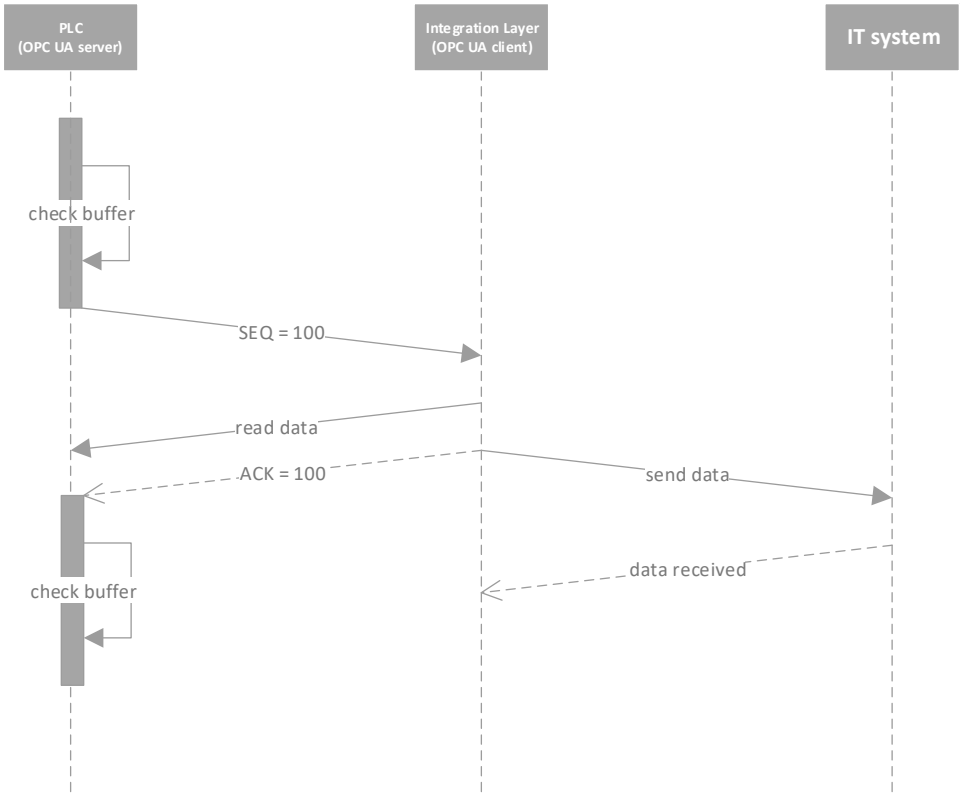


The client checks for available capacity by checking the sequence number (SEQ) and the acknowledge (ACK) number to be equal. If that is the case, the client can directly write its next message to its dedicated interface DB. After the message is written completely, it is sending a sequence number (SEQ) to the OPC UA server to signal completeness. The PLC then saves the message to its internal buffer and verifies if it is ready to receive another message. As soon as that is the case it acknowledges (ACK) the SEQ. Subscribed to the ACK, the client can send the next waiting data to the PLC on change or be ready for a new one.

PLC notifies upper-level system.

The integration layer persistently buffers the PLC data until received by the IT system:

Figure 2-3: PLC notifies upper level system



The PLC checks cyclically if new data is available in its internal buffer. Once data is available, it writes them to the dedicated interface DB and increases the SEQ number node. The OPC UA client is subscribed to the value and gets notified. Once the data is read, the client acknowledges the data transfer.

NOTE

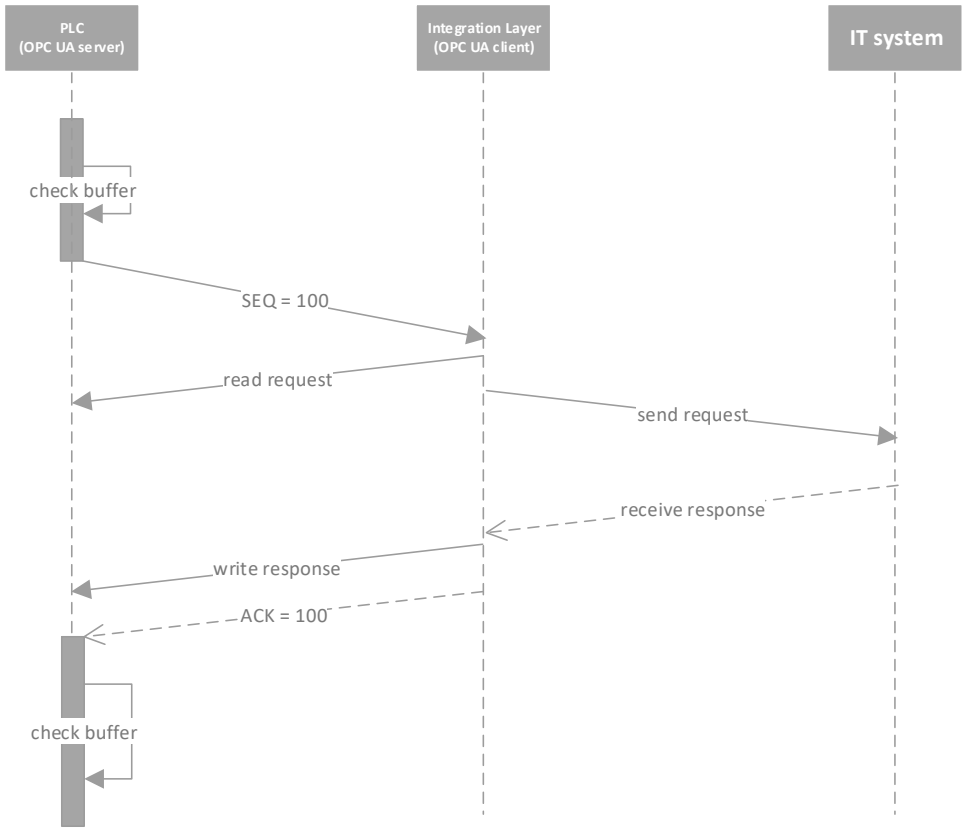
To implement such a message-based data exchange, Siemens also provides an application example for S7-1500 including a corresponding buffer management on PLC. The application example can be used to exchange data as defined in this document. It can be found here: https://support.industry.siemens.com/cs/ww/en/view/109795979

An additional handshake not yet available in the application example was furthermore defined for requesting data from PLC:

PLC requests data from upper-level system.

The integration layer persistently buffers the PLC data until the response is safely sent back to PLC:

Figure 2-4: PLC requests upper-level system



The PLC checks cyclically if new data requests are needed in its internal request buffer. Once a request is available, it increases the SEQ number node. The OPC UA client is subscribed to the value and gets notified. The client can then read the request payload from the provided interface. Once the data is read, the client calls application logic and waits for the response. When the response arrived, it writes the response payload to the response interface of the OPC UA Server and acknowledges the data transfer.

NOTE

To implement such a message-based data exchange, Siemens also provides an application example for S7-1500 including a corresponding buffer management on PLC. The application example can be used to exchange data as defined in this document. It can be found here: https://support.industry.siemens.com/cs/ww/en/view/109795979

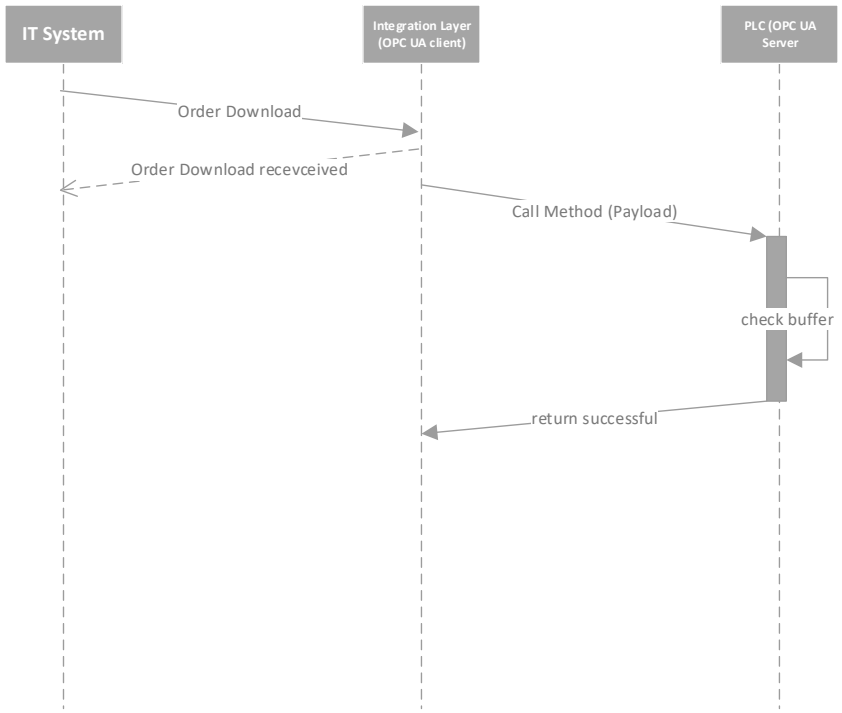
2.3 Handshakes in method-based integration

If your OPC UA Client of the integration layer supports OPC UA method calls, the integration concept is defined as in the following pictures.

Upper-level system sends data to PLC.

The integration layer persistently buffers the data until acknowledged by the PLC:

Figure 2-5: IT system sends data to PLC



Instead of writing data to the dedicated interface DB on the PLC like in section 2.2, a method is called that is taking the data as one or more input parameter and saves it into the internal message buffer. If there was no capability to save it, the method does not succeed.

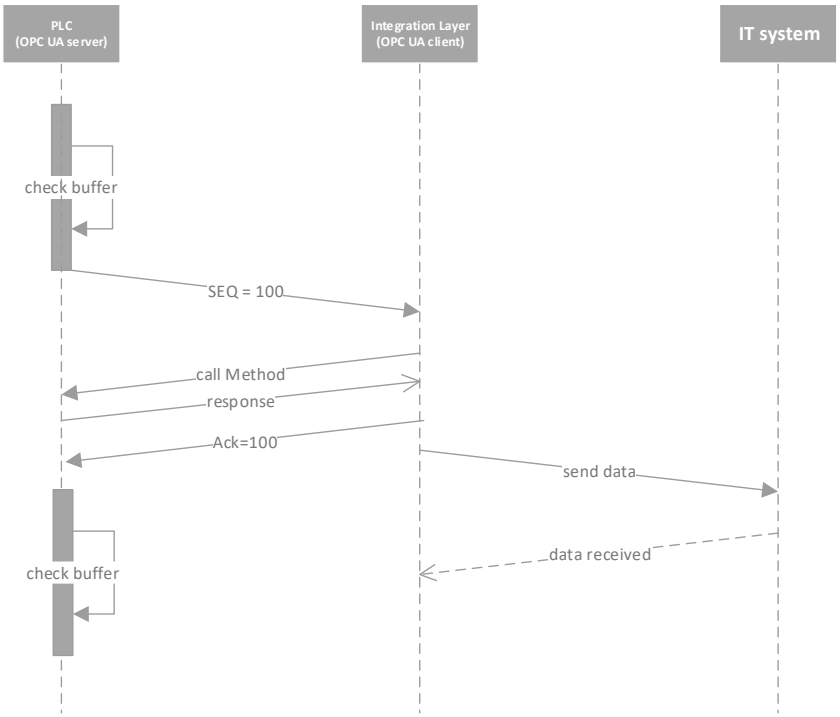
NOTE

To implement such a message-based data exchange, Siemens also provides an application example for S7-1500 including a corresponding buffer management on PLC. The application example can be used to exchange data as defined in this document. It can be found here: https://support.industry.siemens.com/cs/ww/en/view/109795979

PLC notifies upper-level system.

The integration layer persistently buffers the data until successfully forwarded to the IT system. It is recommended to send an acknowledge (ACK) back to signal that data is persistently buffered, to catch the case that the client system crashes between receiving a response and saving the data.

Figure 2-6: PLC notifies upper-level system via method



Instead of reading data from the dedicated interface DB on the PLC like in section 2.2, a method is called that is returning the data as one or more output parameters and saves it within the integration layer.

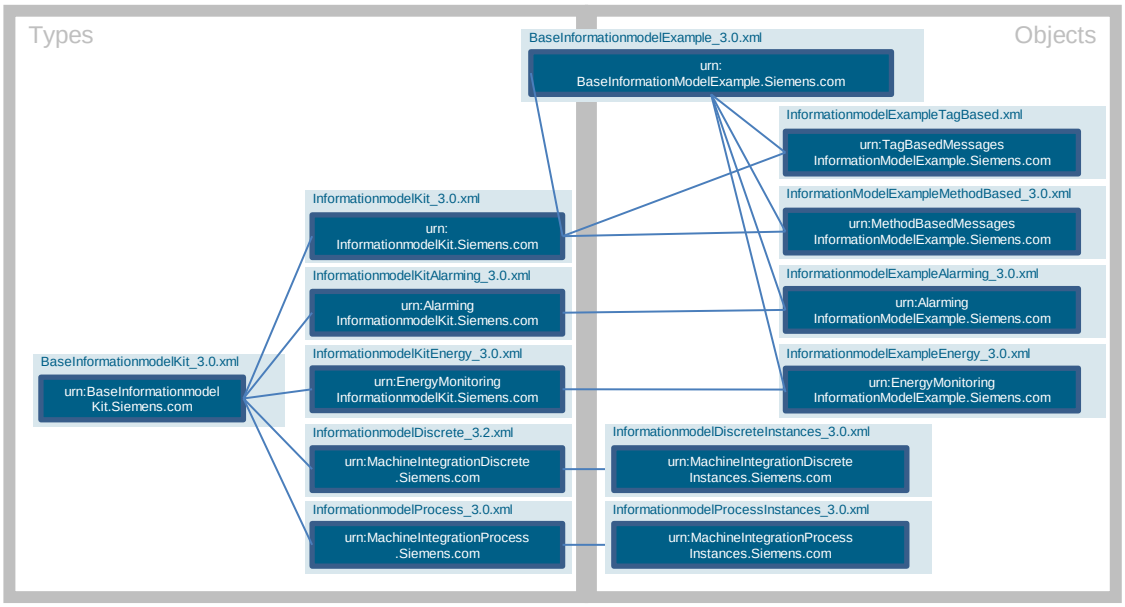
NOTE

To implement such a message-based data exchange, Siemens also provides an application example for S7-1500 including a corresponding buffer management on PLC. The application example can be used to exchange data as defined in this document. It can be found here: https://support.industry.siemens.com/cs/ww/en/view/109795979

2.4 Node sets and namespaces

OPC UA node sets contain types and instances of types. The types and instances can be grouped in namespaces within one node set or can be separated into several namespaces. Commonly namespaces with types and instances are separated and so this information model is structured in this way. Namespaces can separately be imported/exported to node sets and can build up a node set out of different namespaces, where namespaces can have dependencies on each other. All of the provided namespaces are based on the base namespace (BaseInformationmodel). Depending on your use case you can add the required additional namespace that each adds functionality. The following section explains the functionalities, each of the namespaces are providing.

Figure 2-7: Dependencies of the namespaces to each other. Namespaces can be found in the surrounding file and contain objects or types or both.

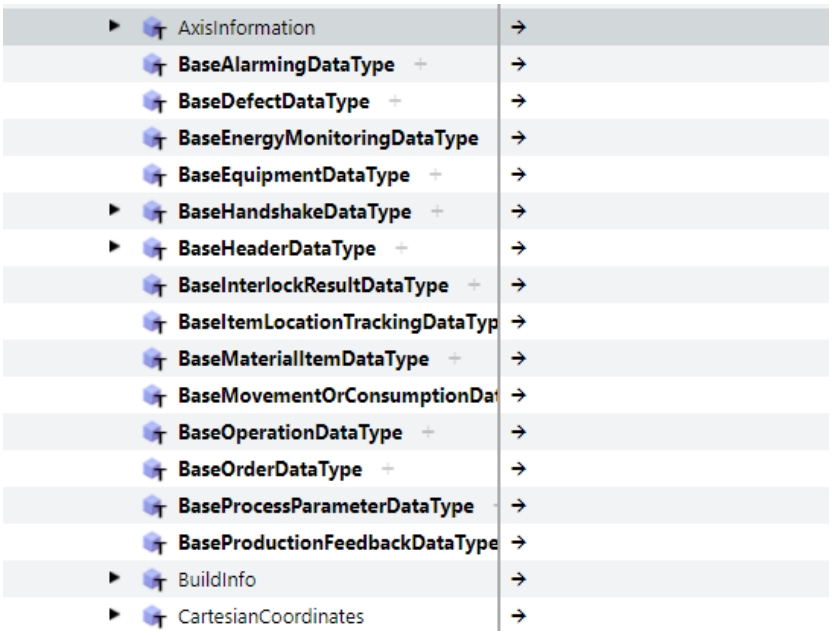


2.4.1 Base information model

The base information model (**urn:BaseInformationmodelKit.Siemens.com**) provides a basic structure for possible additional namespaces. It provides BaseDataTypes or BaseObjectTypes where concrete DataTypes or ObjectTypes can be grouped under. Even if these children of that basic type do not inherit anything from their parents, it is easier to see which different specific types are available in that context or for that usage. Especially if you copy types from each other, as they are still under the same base data type, you can see that they are somehow related to each other. It contains also basic types that should be available for several alternatively imported namespaces, like communication objects.

The base information model is kept short, so that it can be imported in every specific information model without adding unnecessary nodes to the restricted number of a custom PLC server interface. To see the number of possible nodes inside a custom OPC UA server interface, see the related manual of your PLC.

Figure 2-8: Base data types of the "Structure" BaseDataType that contains either basic data types for all the following namespaces or nothing, if they are contained in other namespaces



2.4.2 Information Model Kit

The information model kit (**urn:InformationmodelKit.Siemens.com**) is a toolkit providing Object- and DataTypes which you can use to create your own standardized information model, independently from Siemens products. It was created from our experience and helps you to identify the needed data structure to integrate your specific IT and OT systems and to think of certain use cases and details which you might have forgotten otherwise. The Types can be seen as a first draft that can be edited, copied, removed, or adjusted to represent the data fitting to your process. The Information model Kit currently provides the following production workflow use cases:

Supervision of production equipment

Transparency is important to increase the efficiency and effectiveness of existing and new production lines as well as of machines. Therefore, a consistent recording of production parameters like counters or machine times is required. This is the basis to compare different machines, carry out root cause analyses and to find out potential to improve the production process. Such figures are also known as key performance indicators (KPIs) and can help to fulfill this transparency. Typically, such KPIs based on continuous machine data is visualized in a supervisory system (e.g. SCADA system).

Tracking and tracing of material

The significance of traceability systems has risen due to legislation (e.g. FDA). Many productions in various industries require the documentation of all process values, material consumptions, production steps across the entire production chain to guarantee a full forward and backward genealogy of a product.

Order management

The target of an order management system is to control the production in the most efficient way.

The main functions are:

- Transfer of order data to shopfloor.
- Work in process (WIP) tracking to get runtime feedback of the current progress of the production.
- Operator guidance and electronic work instructions.
- Automatic synchronization of the production process. During the production the order will be enriched with online production data.
- Actual parameter.
- Actual production times / recipe data.

Interlocking and machine control

Purpose of the interlocking is to check any constraints and to check specific steps inside the production. Example is the completeness check of assembled parts at the end of the line. Only in case of positive confirmation the production flow will continue.

Another purpose is to control the machine from a central IT system e.g. stop the machine / line and adjust the speed of the line.

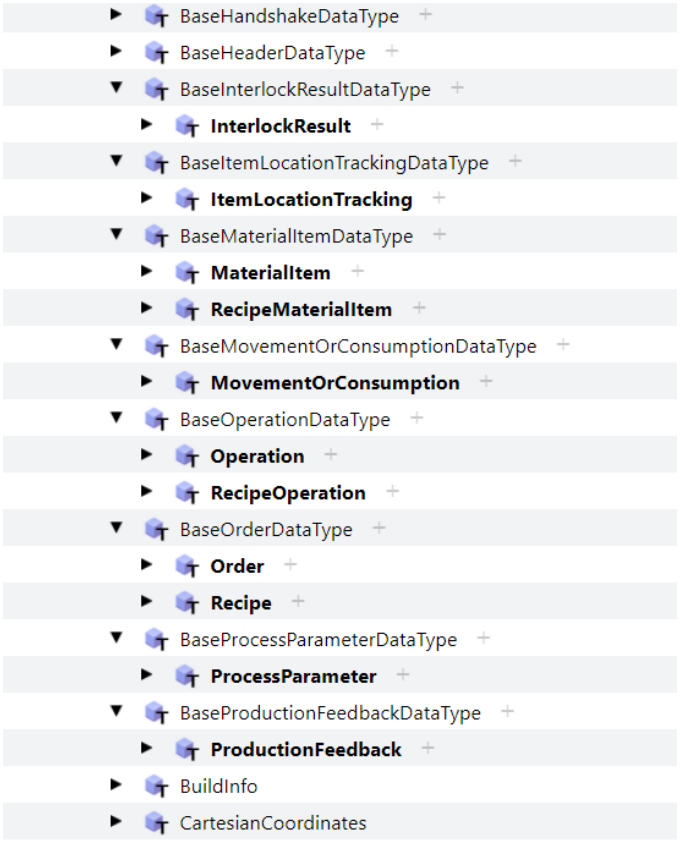
Parameter and recipe management

Machines need set parameter and recipes in order to produce parts. Each product has got their own parameter set.

The following benefits can be achieved by exchanging these data with IT level:

- Central storage and archiving of set parameter / recipe
- Exchange set parameter / recipe from machine to another
- Trigger workflow in Product Lifecycle Management (PLM) in case of Changing parameters

Figure 2-9: Example DataTypes for MOM use cases



2.4.3 Information Model Kit for Alarming

The Types provided for Alarming are not delivered with the standard Informationmodel Kit, as some questions need to be clarified before using them. ProDiag Alarms are warnings, errors or just information that a Siemens PLC produces within its own diagnose system. These alarms can be visualized on panels for the Operator or sent to a process control system to take action. This kind of communication is usually done by internal communication protocols provided by Siemens and does not necessarily need to be published over OPC UA for this functionality. Nevertheless, there can be use cases, where these kinds of alarms should also be forwarded to any other IT systems over OPC UA. Messages on the other hand can contain any structured information independent of the machine. It can but does not need to contain the attributes of an alarm. Messages have a sender and a receiver, and it is important that the information is transferred consistently and sequentially.

Messages sent as OPC UA Alarms (A&C)

When talking about general message-based communication, OPC UA Alarms and Conditions is often mentioned as one of the ways how to transmit messages consistently and completely from OT to IT. OPC UA Alarms can contain and transmit structures and thus eliminate the need of complex message handshake implementation on sender and receiver side. One future use case would be, to define OPC UA Alarms in the PLC and its OPC UA server interface with user data types containing the message content. Unfortunately, this feature is not yet available.

ProDiag Alarms sent as OPC UA Alarms (A&C)

Using OPC UA Alarms and Conditions as communication technology in Client and Server, the ProDiag alarms produced by the PLC can be sent automatically by configuration in the Siemens PLC. The alarm content is then sent as standard alarm structure containing an alarm string to the subscribed client. No changes in the standard server interfaces of the PLC are required nor possible. Therefore, the information model kit is not providing any additional types for this use case. For collecting ProDiag alarms from the PLC within an OPC UA Client, this way is strongly recommended.

ProDiag Alarms sent as messages

Alarms can be seen as messages, that are sent and contain a set of data. Except of using OPC UA A&C, alarms can also be sent as messages using message-based communication. For this, several ways of providing the data are possible. You can either use the provided handshake structures (see section 2.2 or 2.3) if you want to consistently transfer them one by one, which can take a long time if there are many of them. You can also just provide an array of them, overwriting each other, which does not grant consistency, nor completeness, but at least leaves enough time to receive the most recent set of alarms. An application example of this solution can be found here: https://support.industry.siemens.com/cs/ww/en/view/109748168. You can also display only the (monolingual) text of the most recent alarm and subscribe to that, if there is either a maximum of one alarm per publishing interval or only the most recent one is relevant.

Just keep in mind, that for sending alarms over OPC UA, you anyway need to implement logic on PLC side to provide the alarms and that you need to decide between performance and consistency/completeness for event-based communication.

If available, it is recommended to use OPC UA A&C to transfer ProDiag Alarms over OPC UA.

To implement an event-based transmission of PLC alarms over OPC UA, The provided DataType and VariableType in the namespace **urn:AlarmingInformationmodelKit.Siemens.com** offers the standard structure of a ProDiag alarm. It can be used in one of the provided BaseMessageTransferObjectType or just as is.

Figure 2-10: Alarm data type as defined in the PLC for publishing to 3rd IT systems over OPC UA

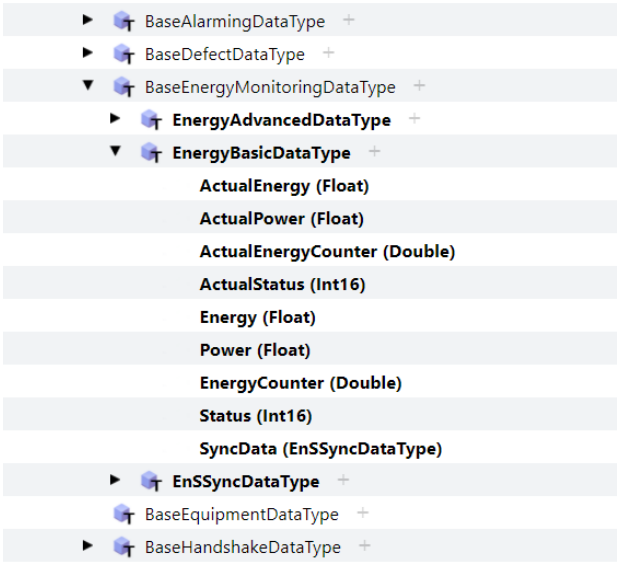
2.4.4 Information Model Kit for Energy Monitoring

The Types provided for Energy monitoring are not delivered with the standard Informationmodel Kit, as some questions need to be clarified before using them.

In general, there are products offered by Siemens to provide Energy Monitoring and Management. You can find documentation of the Energy products of Siemens here: https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10204319 and the necessary libraries (Siemens Energy Suite) here (https://support.industry.siemens.com/cs/ww/en/view/109739102 and here https://support.industry.siemens.com/cs/ww/en/view/109741977). The products provide a holistic concept for Energy monitoring and management and are self-sufficient. They usually don't need to publish energy data over OPC UA for any subscriber but already use an internal communication to implement their features. Nevertheless, there can be use cases, where it can help to additionally forward energy data to any other IT system over OPC UA.

For these use cases the namespace **urn:EnergyMonitoringInformationmodelKit.Siemens.com** provides the two most important user data type used in the referred application examples to be published on the automation system: the basic energy data type (**EnergyBasicDataType**) and the advanced energy data type (**EnergyAdvancedDataType**). For a detailed description of the differences between basic and advanced energy data, refer to the "SIMATIC Energy Suite V17.0" Function Manual, Chapter "Important information on energy data": https://support.industry.siemens.com/cs/ww/en/view/109741977/141708670731

Figure 2-11: Energy data type as defined in the Energy Suite for publishing to 3rd IT systems over OPC UA



2.4.5 Information Model Examples

The Siemens information model kit is supporting to instance an own standardized information model. For those, who want to see an example how it can look like when using the provided Types to instantiate Objects, there are some examples depending on the given environment and conditions.

Generally, the **urn:BaseInformationModelExample.Siemens.com** provides some basic interface components that can be specified independently from the machine type or from further conditions. It contains some structure and basic machine information and status tags. Also, it contains heartbeat tags to evaluate the availability of a communication partner.

Based on that, **urn:AlarmingInformationModelExample.Siemens.com** adds an example, how machine alarms can be collected from the OPC UA interface. It provides a handshake structure (see Chapter 2.2) to collect the alarm data one by one in a tag-based communication without losing one. This is just an example. Also other ways to collect the alarm can be used

If the machine energy data (if implemented as in the out-of-the-box implementation) is additionally required to be published on the OPC UA server interface, an example of how to instantiate the given types is shown in **urn:EnergyMonitoringInformationModelExample.Siemens.com**

urn:TagBasedMessagesInformationModelExample.Siemens.com shows an example for an integration with a tag-based OPC UA Client. It uses the handshake from Chapter 2.2 and the types from **urn:InformationmodelKit.Siemens.com**.

urn:MethodBasedMessagesInformationModelExample.Siemens.com shows an example for an integration with a method-based OPC UA Client. It uses the handshake from Chapter 2.3 and the types from **urn:InformationmodelKit.Siemens.com**.

2.4.6 Information Model Process

One of the most important topics in IT/OT integration is a shopfloor integration of an MES (Manufacturing Execution System). The MES is the closest level above the shopfloor and acts as some kind of a relational data storage of data produced and consumed at shopfloor.

The MES of Siemens for Process Industries is Opcenter Execution Process. From version 4.4 above, it contains a shopfloor integration feature that implements a handshake-based message transfer over OPC UA (with Automation Gateway). The use cases that are implemented out-of-the-box are:

- **Order (Operation) Download**: Downloads a new MES Work Order with all its process information to the PLC.
- **Order (Operation) Status Request**: Notifies the automation system that a status change in MES was requested to be done on automation level.
- **Order (Operation) Status Notification**: Notifies the MES, that a status change was executed on shopfloor (after request or automatically).
- **Material Transaction Execution**: Notifies the MES system, that a material transaction was executed on shopfloor. This can be movements, consumtions, productions or creations.
- Further use cases are planned…

The provided namespace **urn:MachineIntegrationProcess.Siemens.com** contains:

- The Object- and DataTypes to create and extend your own information model to use the out-of-the-box features.

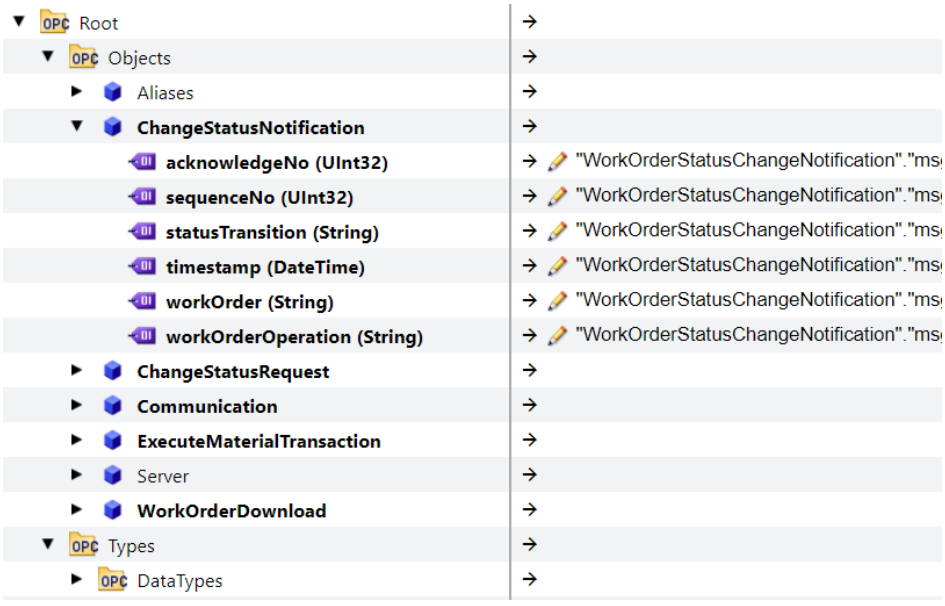
The provided namespace **urn:MachineIntegrationProcessInstances.Siemens.com** contains:

- The instanced and mapped OPC UA server interface which is needed for the out-of-the-box implementation of the published example integration to Opcenter Execution Process. It is using the Object-and DataTypes from **urn:MachineIntegrationProcess.Siemens.com**

For further information on how to integrate Opcenter Execution Process with the provided interface and an example project for TIA Portal, please contact SIMATIC Systems Support (FA S SUP).

As a customer with **Opcenter Execution Process License**, you can view the deliverables and documentation of the shopfloor integration feature at the following link: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204.

Figure 2-12: Interface of communication with Opcenter Execution Process



The provided ObjectTypes are designed in a way that makes mapping in the Automation Gateway configuration as easy as possible. The provided ObjectTypes (and thus the Object instances) are implementing dedicated InterfaceTypes containing the Variables for the handshake structure and from the specific content structure. Unfortunately, the Automation Gateway does not implement VariableTypes, otherwise the payload could be grouped under VariableTypes and be mapped, read and written as a whole Variable.

2.4.7 Information Model Discrete

One of the most important topics in IT/OT integration is a shopfloor integration of an MES (Manufacturing Execution System). The MES is the closest level above the shopfloor and acts as some kind of a relational data storage of data produced and consumed at shopfloor.

The MES of Siemens for Discrete Manufacturing is Opcenter Execution Discrete. From version 4.4 above, it contains a shopfloor integration feature that implements a handshake based message transfer over OPC UA (with Automation Gateway). The use cases that are implemented out-of-the-box are:

- **Save Sequence:** Propagates a just scanned serial number to the following machines in line.
- **Request Next**: Returns the just scanned serial number or the next ordered serial number in a different way (as configured).
- **Request Order Info**: Returns order info to the serial number.
- **Execute Work Order Operation**: Executes a start or stop or abort or… on the operation of the serial number.
- **Request Material Consumption**: sends a request to Opcenter, which materials are to be consumed during the current operation at this equipment.
- **Execute Material Consumption**: Tracks the actual material consumption of material tracking units at this equipment during the current operation in Opcenter
- **Execute Quality Inspection**: trigegrs a configured quality inspection in Opcenter
- **Execute Work Instruction Data Collection**: triggers Opcenter to collect data and save it into work instructions.
- **Execute Item Location Tracking Single**: Sends a change of the current location of an item to Opcenter and whether it is entering, exiting… the equipment
- **Execute Item Location Tracking**: Sends multiple location changes of several items to Opcenter and whether they are entering, exiting… the equipment

The provided namespace **urn: MachineIntegrationDiscrete.Siemens.com** contains.

- The Object- and DataTypes to create and extend your own information model to use the out-of-the-box features.

The provided namespace **urn: MachineIntegrationDiscrete.Siemens.com** contains.

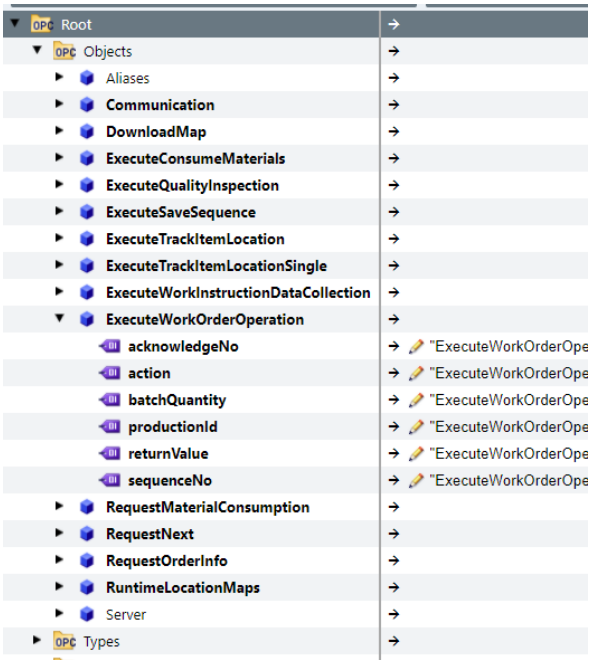
- The instanced and mapped OPC UA server interface which is needed for the out-of-the-box implementation of the published example integration to Opcenter Execution Process. It is using the Object-and DataTypes from **urn:MachineIntegrationDiscrete.Siemens.com**

For further information on how to integrate Opcenter Execution Discrete with the provided interface and an example project for TIA Portal, please contact SIMATIC Systems Support (FA S SUP).

As a customer with **Opcenter Execution Discrete License**, you can view the deliverables and documentation of the shopfloor integration feature at the following link:

https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Figure 2-13: Interface of communication with Opcenter Execution Discrete



The provided ObjectTypes are designed in a way that makes mapping in the Automation Gateway configuration as easy as possible. The provided ObjectTypes (and thus the Object instances) are implementing dedicated InterfaceTypes containing the Variables for the handshake structure, the request content structure and the response content structure. Unfortunately, the Automation Gateway does not implement VariableTypes, otherwise the payload could be grouped under VariableTypes and be mapped, read and written as a whole Variable.

2.5 Usage of OPC UA Types

Before specifying the single types, the choice of (parent) types is explained to introduce the use case types and their purpose.

The Siemens Information Model Kit provides ObjectTypes, DataTypes and VariableTypes to flexibly implement both fields-of-use mentioned in subsection 1.2.2. These types can be instantiated as Objects to be loaded on an OPC UA Server following the own process and own needs.

ObjectTypes

ObjectTypes can be browsed but not directly written/read as a structure through an OPC UA write/read. It can contain other Objects, Variables or Methods. ObjectTypes can either directly have own such children or implement an interface of a BaseInterfaceObjectType and thus have the children of the interface.

VariableTypes on the other hand define a Variable that can also be structured and that can be written as one. In this case all the children inside will be overwritten.

This is the reason why the Information Model Kit uses ObjectTypes for three kinds of objects:

- Objects that contain the tag-based monitoring or control data that is not structured like one big variable, but many single tags changing over time separately. ObjectTypes in SiOME also have the advantage, that only mandatory variables are created during the TIA Portal import and thus no creation of a new DataType is needed for adding or removing members in the instance. These objects are:
  - (Base)EquipmentInformationType
  - (Base)EquipmentStatusType

  Also, there are other communication related Objects that are still providing monitoring or control data:
  - (Base)CommunicationObjectType
- Objects that are implementing different permutations of message transfer objects/variables/methods needed for handshakes and communication debugging (by implementing different BaseInterfaceTypes). These ObjectTypes are subtypes of BaseMessageTransferObjectType and are implementing suggested handshake mechanisms depending on whether methods are supported or not.
  - TagBasedInboundMsgObjectType
  - MethodInboundMsgObjectType
  - TagBasedOutboundMsgObjectType
  - MethodOutboundMsgObjectType
  - TagbasedRequestMsgObjectType

  This ObjectType defines the structure of a message and is used in all of the above BaseMessageTransferObjectTypes to pack payload inside:
  - (Base)EnvelopeObjectType
- Objects needed for the integration of Opcenter Execution Discrete/Process, as the Automation Gateway does not support DataTypes/VariableTypes and structures need to be written tag by tag. These ObjectTypes stored in BaseOpcenterObjectType are
  - (Base)ExecuteMaterialTransaction
  - (Base)StatusChange
  - (Base)WorkOrderDownload
  - (Base)ExecuteSaveSequence
  - (Base)RequestNext
  - (Base)RequestOrderInfo
  - (Base)ExecuteWorkOrderOperation

BaseInterfaceTypes

As mentioned, the message transfer objects are implementing Interface types. The permutations of different children defining one of the suggested handshake mechanisms are defined within the following InterfaceTypes. If an interface is implemented by ("added to") an ObjectType, the ObjectType just takes over its children. Defining specific DataTypes or ObjectTypes in the InterfaceType instead of just using the suggested BaseDataTypes saves time to define them in each object (type) subsequently and can even be easily changed even when Nodes are already created.

- The Siemens Information Model Kit provides different handshake mechanism structures (see above) using these Interfaces:
  - ITagBasedInOrOutMsg
  - IMethodInboundMsg
  - IMethodOutboundMsg
  - ITagbasedRequestMsg

For the same reason the Objects for the Opcenter Execution Discrete/Process integration are built by implementing the interfaces and by that, just adding tags to an ObjectType instead of implementing unsupported VariableTypes. For this the following InterfaceTypes can be found under IOpcenterInterfaceType:

- For Opcenter Execution Process:
  - IHandshake
  - IExecuteMaterialTransaction
  - IWorkOrderDownload
  - IStatusChange
- For Opcenter Execution Discrete:
  - IHandshake
  - IErrorCode
  - IRequest
  - IResponse

DataTypes

DataTypes define the structure, size and format of Variables or Method parameters. Methods, as well as variables, are used in the Siemens Information model Kit to transfer different messages. So the DataTypes define the payload of them.

They are also necessary to tell the TIA Portal import, how to create (and name) the corresponding User Data Type (UDT). This is also the reason why TIA Portal does not create abstract DataTypes, as it cannot know, how much space to allocate. As TIA Portal does not (yet) implement optional and mandatory members of a data type, always all the members are created within the structure, to make sure, all data can be written in case it should be written.

This is also the reason why single items are not excluded from the type by just unselecting them in SiOME. If the DataType cannot be used as-is, it needs to be changed.

- Data types that are supposed to be transmitted within the message payload are:
  - InterlockResult (BaseInterlockResultDataType)
  - ItemLocationTracking (BaseItemLocationTrackingDataType)
  - Order (BaseOrderDataType)
  - Recipe (BaseOrderDataType)
  - ProductionFeedback (BaseProductionFeedbackDataType)
  - MovementOrConsumption (BaseMovementOrConsumptionDataType)
  - Alarming (BaseAlarmingDataType)

- Data types that are used as Variables within the message payload data types, but could also be sent as messages on their own, are:
  - Defect (BaseDefectDataType)
  - MaterialItem (BaseMaterialItemDataType)
  - RecipeMaterialIrem (BaseMaterialItemDataType)
  - Operation (BaseOperationDataType)
  - RecipeOperation (BaseOperationDataType)
  - ProcessParameter (BaseProcessParameterDataType)
- Data types that are specified for the usage within the handshake and envelope structures:
  - MessageHandshake (BaseHandshakeDataType)
  - typeMsgHsk (BaseHandshakeDataType)
  - Header (BaseHeaderDataType)
  - typeMsgHeader (BaseHeaderDataType)
- Others:
  - EnergyAdvandedDataType (BaseEnergyMonitoringDataType)
  - EnergyBasicDataType (BaseEnergyMonitoringDataType)

VariableTypes

The VariableTypes are created according to the Datatypes, so that every DataType has a VariableType to be sent not only within a method parameter, but also by writing or reading that structure within the handshake mechanism (or somehow else). When changing the DataType, also the VariableType needs to be recreated, in case it is used within the handshake mechanism.

Parent types

As DataTypes and VariableTypes require a change in many cases while it might be necessary to keep the original type present, it happens, that types concerning the same use case or thing are duplicated. To keep the overview of these similar types, they are grouped under one base parent type. It also eases/restricts the choice of possible Data- or ObjectTypes when the abstract type is already specified.

3 Information Model

3.1 Information object

3.1.1 EquipmentInformationObjectType ()

ObjectTypes > BaseObjectType > BaseEquipmentInformationObjectType

The EquipmentInformationObjectType defines an ObjectType providing basic information about the equipment, like the equipment name or the equipment serial number. This object shall be used e.g., to identify the equipment and to verify which version of the interface is available at the OPC UA interface. It is usually filled statically.

Table 3-1

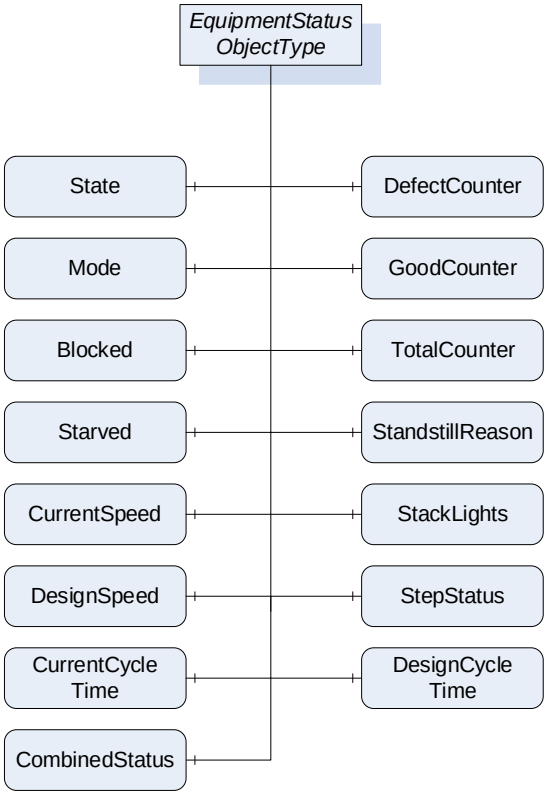
Name	TypeDefinition	DataType	Description
InterfaceVersion Vendor	PropertyType	String	Vendor specific version of the interface: syntax Vxx.xx.xxx
InterfaceVersion Project	PropertyType	String	Project specific version of the interface: syntax Vxx.xx.xxx
EquipmentVersion	PropertyType	String	Vendor specific version of the equipment: syntax Vxx.xx.xxx
EquipmentSerialNo	PropertyType	String	Unique serial number of the equipment (vendor specific)
EquipmentName	PropertyType	String	Equipment name by the equipment vendor (vendor specific)
EquipmentTypeID	PropertyType	Int32	Equipment type classification e.g. packaging machine
AreaID	PropertyType	Int32	Area ID for all equipment in the plant / plant specific area identifier (defined by the project owner, ID can be written at commissioning)
LineID	PropertyType	Int32	Line ID for all equipment in the plant / plant specific line identifier (defined by the project owner, ID can be written at commissioning)
EquipmentModuleID	PropertyType	Int32	Sub equipment ID for a modules of a modular production equipment, is used only for modular equipment if the single modules need to be identified e.g. left or right press
EquipmentCode	PropertyType	String	Unique Equipment name/code - PIC (Plant Identification Code) to identify a single machine
VendorID	PropertyType	Int32	Unique vendor-supplier identifier of the equipment vendor
EquipmentID	PropertyType	Int32	Unique ID for all equipment in the plant / plant specific equipment identifier (defined by the project owner, ID can be written at commissioning

3.2 Status object

3.2.1 EquipmentStatusObjectType

urn:KitInformationmodel.Siemens.com
ObjectTypes > BaseObjectType > BaseEquipmentStatusObjectType

Figure 3-1: EquipmentStatusObjectType



The EquipmentStatusObjectType defines an ObjectType that can be used to gather machine status relevant information. Besides that, the object also provides basic KPI information like counter or standstill reasons. Compared to the information object the members here are dynamically changing over time.

Table 3-2

Name	TypeDefinition	DataType	Description
State	BaseDataVariableType	MachineStateEnumType	provides the status of the machine. e.g. execute. This variable should not be used if you have an additional state machine as defined e.g. in PackML
Mode	BaseDataVariableType	MachineModeEnumType	provides the mode of the machine e.g. production. This variable should not be used if you have an additional state machine.
Combined Status	BaseDataVariableType	EquipmentCombinedStatusEnumType	contains status as well as mode of the machine. Can be extended with further status as well as standstill reasons to support OEE applications which can only manage one machine status integer.
Blocked	BaseDataVariableType	Boolean	machine suspends because the equipment does not get rid of outbound material

Name	TypeDefinition	DataType	Description
Starved	BaseDataVariableType	Boolean	machine suspends because the equipment is missing inbound material
CurrentSpeed	BaseDataVariableType	Float	provides the current speed of the machine (e.g. pcs/s)
DesignSpeed	BaseDataVariableType	Float	provides the design speed of a machine (e.g. pcs/s)
Designt CycleTime	BaseDataVariableType	Float	provides the design cycle time of a machine (e.g. s/pcs)
Current CycleTime	BaseDataVariableType	Float	provides the current cycle time of a machine (e.g. s/pcs)
DefectCounter	BaseDataVariableType	UInt64	provides a counter for defective parts
GoodCounter	BaseDataVariableType	UInt64	provides a counter for good parts
TotalCounter	BaseDataVariableType	UInt64	total counter of machine (good + defective counter)
Standstill Reason	BaseDataVariableType	MachineStandstillReasonEnumType	provides the stop reason of the machine
StackLight	StackLightsFolderType		provides the stack light of the machine
StepStatus	BaseDataVariableType	MachineStepStatusEnumType	provides the detail step status of the machine

3.2.2 StackLightsFolderType

urn:KitInformationmodel.Siemens.com
ObjectTypes > BaseObjectType > FolderType

The StackLightFolderType contains the stacked lights of the stacklights as objects. They are represented as MandatoryPaceholders of an arbitrary number.

Table 3-3

Name	TypeDefinition	DataType	Description
Stacklights<no>	StackLightObjectType		provides a set of stacked lights

3.2.3 StackLightObjectType

urn:KitInformationmodel.Siemens.com
ObjectTypes > BaseObjectType > BaseStacklightObjectType

The StackLightObjectType defines an ObjectType that can be used to illustrate a single light of the machine stack light. The object can be instantiated multiple times in the StackLights folder to display the complete stack light of the machine.

Each object contains the following members:

Table 3-4

Name	TypeDefinition	DataType	Description
Position	PropertyType	Int32	stack light position from bottom to top, starting with 1
Color	BaseData VariableType	StackLightColorEnumType	color of the single light of the stack light
Behaviour	BaseData VariableType	StackLightBehaviorEnumType	Describes predefined modes of a light. If any behavior, e.g. fast/ slow flashing is missing it is possible to extend the enumeration
Meaning	BaseData VariableType	String	describes the meaning of the color and behaviour combination e.g. a flashing red light could be an emergency stop

3.2.4 MachineStateEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

The MachineStateEnumType describes predefined states for a machine. This is a default state enumeration. It is possible to extend it for your project specific use-cases.

Table 3-5

Value	Description
Undefined_0	Undefined
Clearing_1	The machine is in the time period between the release after an Emergency Stop by the operator and the resulting end state
Stopped_2	The machine has electrical power but is in a stationary state.
Starting_3	The machine is in production start. This is not to be confused with program Start up. An example might be the self-test of a packing robot, which tests the functionality of the individual servo motors through a movement test.
Idle_4	The machine is ready to carry out its intended function. It is however in a waiting state and must first be brought into operation by the operators or by external automatic release.
Suspended_5	Will be used in case of no detail status of suspended is needed
Execute_6	The machine is carrying out its intended function
Stopping_7	The machine is being transferred to the stopped state (Stopped) by a controlled stop routine.
Aborting_8	The machine is in the time period between the occurrence of the failure and the resulting end state (Emergency Stop).
Aborted_9	Machine is aborted and is not operating any more.
Holding_10	The machine is in the time period between the holding and the resulting end state (Held, Equipment Failure, External Failure).
Held_11	The machine is not carrying out its intended function due to an unpermitted deviation from the SET-state detected by a control system sensor.
Unholding_12	The machine is in the time period between the release of held (Held, Equipment Failure, External Failure) by the operator and the resulting end state (Operating).
Suspending_13	The machine is in the time period between the initiation of Suspending and the resulting end state (Prepared, Lack, Tailback, Lack Branch Line or Tailback Branch Line).
Unsuspending_14	The machine is in the time period between the release of being suspended (Prepared, Lack, Tailback, Lack Branch Line or Tailback Branch Line) and the resulting end state (Operating).

Value	Description
Resetting_15	The machine is in the time period between the release after a stop (Stopped) and the resulting end state (Idle).
Completing_16	Order quantity is going to reached.
Complete_17	Order quantity is reached.
Setup_18	Setup activities are performed on the machine.
EmergencyStop_19	The machine is in EmergencyStop state after operator hat pushed the emergency stop
HeldEquipment Failure_20	Failure, which occurs in the machine itself and leads to a machine stop (in accordance with DIN 8782).
HeldExternal Failure_21	A failure which is not attributable to the machine, but which nonetheless leads to a machine stop (in accordance with DIN 8782).
Suspended Prepared_22	The machine is ready to carry out its intended function. However, it is in a waiting state which was not recognized as a lack or tailback state and can automatically start again.
Suspended Lack_23	The machine is not carrying out its intended function due to a lack detected by the sensor system in the inlet of the machine (Machine Stop).
Suspended Tailback_24	The machine is not carrying out its intended function due to a tailback detected by the sensor system in the outlet of the machine (Machine Stop).
SuspendedLack BranchLine_25	The machine is not carrying out its intended function due to a lack in the secondary inlet or a tailback in the secondary outlet of the machine (Machine Stop).

3.2.5 MachineModeEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

The MachineModeEnumType describes predefined modes for a machine. This is a default mode enumeration. It is possible to extend it for your project specific use-cases.

Table 3-6

Value	Description
Undefined_0	Undefined
Produce_1	The machine executes relevant logic in production mode. The machine is producing product or able to produce product.
Maintenance_2	This mode indicates that the machine is in maintenance mode. This mode would typically be used for faultfinding, machine trials or testing operational improvements.
Manual_3	This mode indicates that the machine is in manual mode. This feature may be used for the commissioning of the individual drives, verifying the operation of synchronized drives, testing the drive as result of modifying parameters, etc.
SemiAutomatic_4	The machine executes relevant logic in semi-automatic mode. The machine is producing product or able to produce product where some operations are executed manually, while other operations are executed automatic.
Remote_5	The machine executes relevant logic in remote mode, e.g. it is controlled from a supervision control system.
StepMode_6	The machine executes relevant logic in step mode. This mode is typically used during commissioning where the machine processes a product step by step and the operator must start the next step manually.

3.2.6 MachineCombinedStatusEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

The MachineCombinedStatusEnumType describes status as well as mode of the machine. It can be extended with further status as well as standstill reasons to support OEE applications which can only manage one machine status integer.

Table 3-7

Value	Description
GeneralError_1	General Error
Production_2	Production
Waiting_3	Waiting (e.g. between two production steps)
Blocked_4	Blocked
Idle_5	Idle (e.g. no further production order)
Undefined_6	Undefined
Maintenance_7	Maintenance.

3.2.7 MachineStandstillReasonEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

The MachineStandstillReasonEnum should contain all the machine specific standstill reasons. It is necessary to add all machine specific standstill reasons. The following table is just a suggestion how to cluster different category. Detailed values and descriptions must be defined at project level for each machine type. The number of the value is the start value of the range. Each range ends with the start number of the next range, e.g.
MachineInternalSafeties_2-23
MachineInternalOperatorAction_33-64

Table 3-8

Value	Description
Undefined_0	Undefined
MachineInternal EStop_1	Machine Internal EStop
MachineInternal Safeties_2	Safety relevant standstill reason
MachineInternal OperatorAction_33	Standstill due to operator action
MachineInternal ProductAndMaterial Related_65	Standstill due to product or material reasons
MachineInternal EquipmentFault_257	Machine internal equipment fault
MachineInternal AllOtherFaults_513	Machine internal all other faults
MachineExtUpstream ProcessReason_2000	Machine ext. upstream process reason main product flow
MachineExtDownstream ProcessReason_3000	Machine ext. downstream process reason main product flow
OutOfService_4000	Out of service (planned and unplanned)

Value	Description
OtherExternal Reasons_4500	Other external reasons branch- or sub-utility equipment

3.2.8 StackLightColorEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

The StackLightColorEnumType describes predefined colors of a stack light. If a color is missing it is possible to extend the enumeration for the project.

Table 3-9

Value	Description
Undefined_0	Undefined
Red_1	Red
Yellow_2	Yellow
Green_3	Green
Blue_4	Blue
White_5	White

3.2.9 StackLightBehaviorEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

The StackLightBehaviourEnumType describes predefined modes of a light. If any behaviour, e.g. fast/ slow flashing is missing it is possible to extend the enumeration.

Table 3-10

Value	Description
Undefined_0	Undefined
OFF_1	OFF
Solid_2A	Solid
Flashing_3	Flashing

3.2.10 EnergyAdvancedType

urn:KitInformationmodelEnergyMonitoring.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseEnergyMonitoringType

This VariableType is created from the DataType

EnergyAdvancedDataType

urn:KitInformationmodelEnergyMonitoring.Siemens.com
DataTypes > BaseDataType > Structure > BaseEnergyMonitoringDataType

It contains energy monitoring not needed by, but already present and implemented on the automation system by usage of the Energy Suite (see subsection 2.4.4).

Advanced energy data is energy data relevant to visualization. This includes:

- Phase-specific current, voltage, power and power factor values
- Total power, total energy and total power factor values
- Frequency values

Table 3-11

Name	Type	Description
Voltage1N	Float	Momentary voltage between phase 1 and N
Voltage2N	Float	Momentary voltage between phase 2 and N
Voltage3N	Float	Momentary voltage between phase 3 and N
Voltage12	Float	Momentary voltage between phase 1 and 2
Voltage23	Float	Momentary voltage between phase 2 and 3
Voltage31	Float	Momentary voltage between phase 3 and 1
Current1	Float	Momentary current at phase 1
Current2	Float	Momentary current at phase 2
Current3	Float	Momentary current at phase 3
Frequency	Float	Momentary frequency
PowerFactor1	Float	Current power factor of Phase 1
PowerFactor2	Float	Current power factor of Phase 2
PowerFactor3	Float	Current power factor of Phase 3
TotalPowerFactor	Float	Momentary total power factor
ApparentPower1	Float	Current apparent power of Phase 1
ApparentPower2	Float	Current apparent power of Phase 2
ApparentPower3	Float	Current apparent power of Phase 3
TotalApparentPower	Float	Momentary total apparent power
ActivePower1	Float	Current active power of Phase 1
ActivePower2	Float	Current active power of Phase 2
ActivePower3	Float	Current active power of Phase 3
TotalActivePower	Float	Momentary total active power
ReactivePower1	Float	Current reactive power of Phase 1
ReactivePower2	Float	Current reactive power of Phase 2
ReactivePower3	Float	Current reactive power of Phase 3
TotalReactivePower	Float	Momentary total reactive power
TotalApparentEnergy	Double	Momentary total apparent energy
TotalActiveEnergy	Double	Momentary total active energy
TotalReactiveEnergy	Double	Momentary total reactive energy
Status	Int16	Status of the extended data

3.2.11 EnergyBasicType

urn:KitInformationmodelEnergyMonitoring.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseEnergyMonitoringType

This VariableType is created from the DataType

EnergyBasicDataType

urn:KitInformationmodelEnergyMonitoring.Siemens.com
DataTypes > BaseDataType > Structure > BaseEnergyMonitoringDataType

It contains energy monitoring not needed by, but already present and implemented on the automation system by usage of the Energy Suite (see subsection 2.4.4).

Basic energy data is energy data relevant to accounting:

- Periodically calculated energy consumption
- Average output per archiving period
- Current value of the hardware counter (energy count)

Table 3-12

Name	Type	Description
ActualEnergy	Float	Momentary energy value
ActualPower	Float	Momentary power value
ActualEnergy Counter	Double	Current energy counter value
ActualStatus	Int16	Status of actual values
Energy	Float	Cumulative energy value at the end of the archiving period
Power	Float	Average power value at the end of the archiving period
EnergyCounter	Double	Energy counter value at the end of the archiving period
Status	Int16	Status of the cumulative energy and average power
SyncData	EnSSyncDataType	Synchronization data

3.2.12 EnSSyncType

urn:KitInformationmodelEnergyMonitoring.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseEnergyMonitoringType

This VariableType is created from the DataType

EnSSyncDataType

urn:KitInformationmodelEnergyMonitoring.Siemens.com
DataTypes > BaseDataType > Structure > BaseEnergyMonitoringDataType

It contains synchronization data for the EnergyBasic(Data)Type (see subsection 2.4.4).

Table 3-13

Name	Type	Description
SyncPeriod	Float	Synchronisation period
SyncTime	Float	Synchronisation time
SyncPulse	Boolean	Synchronisation pulse
SyncTimeStamp	DateTime	Synchronisation time stamp
Error	Boolean	Error bit
Status	Int16	Status code

3.3 Message structure objects

3.3.1 BaseMessageTransferObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType

The BaseMessageTransferObjectType defines a BaseObjectType that contains different ObjectTypes each for publishing or receiving OPC UA messages or requests in a consistent way, by either using a handshake mechanism or methods. It has Subtypes that implement InterfaceTypes

3.3.2 MethodInboundMsgObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseMessageTransferObjectType

The MethodInboundMsgObjectType is used for receiving any message content consistently as a server with the help of OPC UA methods. The ObjectType implements the InterfaceType IMethodInboundMsg and inherits all the members of it. For further description see the corresponding InterfaceType.

3.3.3 MethodOutboundMsgObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseMessageTransferObjectType

The MethodOutboundMsgObjectType is used for sending any message content consistently as a server with the help of OPC UA methods. The ObjectType implements the InterfaceType IMethodOutboundMsg and inherits all the members of it. For further description see the corresponding InterfaceType.

3.3.4 TagbasedInboundMsgObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseMessageTransferObjectType

The TagbasedInboundMsgObjectType is used for receiving any message content consistently as a server using tag-based communication. The ObjectType implements the InterfaceType ITagbasedInOrOutMsg and inherits all the members of it. For further description see the corresponding InterfaceType.

3.3.5 TagbasedOutboundMsgObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseMessageTransferObjectType

The TagbasedOutboundMsgObjectType is used for sending any message content consistently as a server using tag-based communication. The ObjectType implements the InterfaceType ITagbasedInOrOutMsg and inherits all the members of it. For further description see the corresponding InterfaceType.

3.3.6 TagbasedRequestMsgObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseMessageTransferObjectType

The TagbasedOutboundMsgObjectType is used for sending any request message content consistently and receiving any response message content consistently as a server using tag-based communication. The ObjectType implements the InterfaceType ITagbasedRequestMsg and inherits all the members of it. For further description see the corresponding InterfaceType.

3.3.7 IMethodInboundMsg

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseInterfaceType

According to the handshake described in 2.3 the message structure just needs a method to be called that transfers inbound parameters. For this the interface contains the following members:

Table 3-14

Name	TypeDefinition	DataType	Description
SetEnvelope	-	-	The method can be called from the upper-level system to send the content and header in a consistent way

3.3.8 IMethodOutboundMsg

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseInterfaceType

According to the handshake described in 2.3 the message structure needs a method to be called that responds with return values and a handshake to start and complete data transmission. For this the interface contains the following members:

Table 3-15

Name	TypeDefinition	DataType	Description
GetEnvelope	-	-	The method can be called from the upper-level system to receive the content and header in a consistent way
Handshake	BaseHandshake Type	BaseHandshake DataType	The handshake object to initiate and acknowledge the message transfer (to be specified before used)

3.3.9 ITagbasedInOrOutMsg

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseInterfaceType

According to the handshake described in 2.2 and 2.3 the message structure needs an Envelope that contains the message and a handshake to start and complete data transmission. In case of the inbound message handshake, the envelope gets written by the upper-level system and needs to be saved in the buffer after the sequence number in the handshake is triggered, in case of an outbound message, the envelope gets read by the upper-level system as soon as the sequence number is triggered. For both the interface contains the following members:

Table 3-16

Name	TypeDefinition	DataType	Description
Envelope	EnvelopeObjectType	-	Consists of a predefined header and the actual content and can be read or written by the upper-level system to send or receive the messagein a consistent way
Handshake	BaseHandshake Type	BaseHandshake DataType	The handshake object to initiate and acknowledge the message transfer (to be specified before used)

3.3.10 ITagbasedRequestMsg

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseInterfaceType

According to the handshake described in 2.2 and 2.3 the message structure needs an Envelope that contains the request message (to be read from the upper-level system), an Envelope that contains the response message (to be written by the upper-level system) and a handshake to start and complete data transmission. Also, there is an optional ErrorCode added, that can display protocol related issues to handle independently from application data. For this the interface contains the following members:

Table 3-17

Name	TypeDefinition	DataType	Description
Request Envelope	EnvelopeObjectType	-	Consists of a predefined header and the actual content of the request message.
Response Envelope	EnvelopeObjectType	-	Consists of a predefined header and the actual content of the response message.
ErrorCode	BaseData VariableType	Int16	Code that can display protocol related issues to handle independently from application data
Handshake	BaseHandshake Type	BaseHandshake DataType	Handshake object to exchange data in a consistent way with an upper level system (to be specified before used)

3.3.11 EnvelopeObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseEnvelopeObjectType

The EnvelopeObjectType defines an ObjectType to publish OPC UA messages which consist of a predefined header and the actual content. The header is used mainly for debugging and tracing purposes. The content is abstract to contain any of the provided or own types.

It contains these members:

Table 3-18

Name	TypeDefinition	DataType	Description
Content	BaseData VariableType	Base DataType	Variable which contains the payload of your message. This can be any OPC UA data type
Header	BaseHeader Type	BaseHeader DataType	contains header information of your message (to be specified before used)

3.3.12 GetEnvelope

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseInterfaceType > IMethodOutboundMsg

```
GetEnvelope (
          [out] BaseDataType Content
          [out] BaseHeaderDataType Header
             );
```

This method can be called by the upper-level system and returns the message that is just to be sent by the automation system, which consists of a header and a content.

It expects the following parameters:

Table 3-19

Argument	Type	Description
Content	Base DataType	Received content from the machine (choose any DataType)
Header	BaseHeader DataType	Basic header information of the message (to be specified before used)

And returns the following result codes:

Table 3-20

Result Code	Description
Bad_NotImplemented	Requested operation is not implemented

3.3.13 SetEnvelope

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseInterfaceType > IMethodInboundMsg

```
SetEnvelope (
          [in]  BaseDataType content
          [in]  BaseHeaderDataType header
          [out] Boolean successful
                );
```

This method can be called by the upper-level system and hands over the message that is just to be sent by the automation system, which consists of a header and a content.

It expects the following parameters:

Table 3-21

Argument	Type	Description
Content	Base DataType	Content you want to send (choose any DataType)
Header	BaseHeader DataType	Basic header information of the message (to be specified before used)
successful	Boolean	Indicates that the call was successful, and the message could be saved

And returns the following result codes:

Table 3-22

Result Code	Description
Bad_NotImplemented	Requested operation is not implemented
Bad_ArgumentsMissing	The client did not specify all the input arguments
Bad_InvalidArgument	Used to indicate in the operation level results that one or more of the input arguments are invalid

3.3.14 HeaderType

urn:BaseInformationmodelkit.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseHeaderType

The header type can be used for any kind of telegrams. It includes general information to transfer the actual content. It follows OPC UA specification naming convention. It contains the following members:

Table 3-23

Name	DataType	Description
Id	UInt32	Unique message id for a message. Generated by the source. Is increasing for each message sent within sequence even across interfaces. Is unequal to the sequence number of the handshake
Timestamp	DateTime	Timestamp from the sender when message was created
Type	String	Type of message or different additional attribute

3.3.15 typeMsgHeaderType

urn:BaseInformationmodelkit.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseHeaderType

The header type can be used for any kind of telegrams. It includes general information to transfer the actual content. It follows the Siemens style guide for implementing STEP7 projects and can therefore be directly imported to TIA Project to implement the suggested handshake interface from the application example (https://support.industry.siemens.com/cs/ww/en/view/109795979).

It contains the following members:

Table 3-24

Name	DataType	Description
uId	UInt32	Unique message id for a message. Generated by the source. Is increasing for each message sent within sequence even across interfaces. Is unequal to the sequence number of the handshake
timestamp	DateTime	Timestamp from the sender when message was created
type	String	Type of message or different additional attribute

3.3.16 MessageHandshakeType

urn:BaseInformationmodelkit.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseHandshakeType

Basic structure to exchange handshake information. It follows OPC UA specification naming convention. It contains the following members:

Table 3-25

Name	DataType	Description
SequenceNo	UInt32	Unique sequence number for this handshake interface to initiate the communication process
AcknowledgeNo	UInt32	Unique acknowledge number for this handshake interface, to acknowledge a communication process

3.3.17 typeMsgHskType

urn:BaseInformationmodelkit.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseHandshakeType

Basic structure to exchange handshake information. It follows the Siemens style guide for implementing STEP7 projects and can therefore be directly imported to TIA Project to implement the suggested handshake interface from the application example (https://support.industry.siemens.com/cs/ww/en/view/109795979).

It contains the following members:

Table 3-26

Name	DataType	Description
seq	UInt32	Unique sequence number for this handshake interface to initiate the communication process
ack	UInt32	Unique acknowledge number for this handshake interface, to acknowledge a communication process
hasTimeout	Boolean	Will be true after handshake is not acknowledged for a predefined time

3.4 Message content types

3.4.1 ItemLocationTrackingType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseItemLocationTrackingType

This VariableType is created from the DataType

ItemLocationTracking

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseItemLocationTrackingDataType

This data type is used to track the physical location of a specific item (e.g. a finished part, handling unit, tool) in the upper-level system. The location can be a logical one (e.g. a station id, area id) or in coordinates.

Geographic coordinate reference systems

- WGS-84: longitude, latitude (x, y)
- WGS-84-3D: longitude, latitude, height (x, y, z)

Cartesian coordinate reference systems

- Cartesian: x, y
- Cartesian 3D: x, y, z

It contains the following members:

Table 3-27

Name	DataType	Description
SerialNo	String	Unique Id of the item
LocationId	String	Id which identifies the location e.g. equipment id
Timestamp	DateTime	Timestamp of the physical event
Interlock	Boolean	Execute an interlock check on MES level
LocationTracking	LocationTrackingEnumType	Type of location
Xposition	Int32	X-Position of workpiece in mm
Yposition	Int32	Y-Position of workpiece in mm
Zposition	Int32	Z-Position of workpiece in mm
MaterialId	String	Identifier of the material
BatchId	String	Identifier of the Batch

3.4.2 ProductionFeedbackType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseProductionFeedbackType

This VariableType is created from the DataType

ProductionFeedback

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseProductionFeedbackDataType

It summarizes all main information about the production progress (of more or less an order with its status). This information typically is reported at the end of an operation or in case of the order is finished. You can use substructure of operation in order to transfer data like process parameters and used materials. It contains the following members:

Table 3-28

Name	Type	Description
MaterialId	String	Material number of the finished product
SerialNo	String	Serial number of the finished product
Quantity	Number	Produced quantity as int or float (needs to be specified when instantiated)
OrderId	String	Id of the order number
BatchId	String	Id of the batch
SequenceId	String	Unique Id which represents in which order the product should be produced
EquipmentId	String	Id if the equipment
Timestamp	DateTime	Timestamp when parts were produced
Operations	Operation[]	Array of operations
QualityStatus	QualityStatusEnumType	Product quality
Defects	DefectType	Array of defects

3.4.3 OrderType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseOrderType

This VariableType is created from the DataType

Order

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseOrderDataType

This data structure is used to send order data to the shop floor (or back).

Table 3-29

Name	Type	Description
MaterialNo	String	Target material number
SerialNo	String	Target serial number
Quantity	Number	Target quantity as int or float (needs to be specified)
OrderId	String	Id of the order
BatchId	String	Id of the batch
SequenceId	String	Sequence of the produced item in the production line

Name	Type	Description
EquipmentId	String	Id of the equipment (equals unit in batch id) or equipment group
Timestamp	DateTime	Timestamp
Operations	OperationType []	Array of operations which need to be executed and how
Defects	DefectType	Array of already known defects of the product (e.g. scratches in the varnish)
OrderStatus	OrderStatusEnumType	Status of the order

3.4.4 RecipeType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseOrderType

This VariableType is created from the DataType

Recipe

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseOrderDataType

This data structure is used to send recipe data to the shop floor (or back).

Table 3-30

Name	Type	Description
MaterialNo	String	Target material number
Quantity	Number	Target quantity as int or float
RecipeId	String	Id of the recipe
RecipeStatus	String	Status of the recipe
Revision	Strin	Revision of the recipe
EquipmentId	String	Id of the equipment (equals unit in batch id) or equipment group
Timestamp	DateTime	Timestamp
Operations	OperationType[]	Array of operations which need to be executed and how

3.4.5 MovementOrConsumptionType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseMovementOrConsumptionType

This VariableType is created from the DataType

MovementOrConsumption

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseMovementOrConsumptionDataType

It contains a material movement or consumption to the IT system for i.e. tracking and tracing purposes. It can represent a mixing, transportation, creation, or production transaction. It contains the following members:

Table 3-31

Name	TypeDefinition	Type	Description
SourceLocation	BaseData VariableType	String	Identifier of the source equipment or location
SourceMaterial	BaseData VariableType	MaterialItemType	Material Id, serialnumber, quantity or anything else to identify and describe the source part, batch or material
Destination Location	BaseData VariableType	String	Identifier of the destination equipment or location
Destination Material	BaseData VariableType	MaterialItemType	Material Id, serialnumber, quantity or anything else to identify and describe the source part, batch or material
Order	BaseData VariableType	OrderType	The order in which contest this movement or consumtion happens
Timestamp	BaseData VariableType	DateTime	Timestamp

3.4.6 InterlockResultType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseInterlockResultType

This VariableType is created from the DataType

InterlockResult

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseInterlockResultDataType

This type contains the result of the interlock check in the IT system. It contains the following members:

Table 3-32

Name	TypeDefinition	Type	Description
SerialNo	BaseData VariableType	String	Unique id of the final material (could also be a handling unit id)
LocationId	BaseData VariableType	String	Location id e.g. the equipment id, where it is released or not.
Timestamp	BaseData VariableType	DateTime	Timestamp
InterlockRelease	BaseData VariableType	Boolean	Result of the interlock check (e.g. true indicates the interlock is released)
LockReason	BaseData VariableType	String	Reason why interlock is not released
Operation	BaseData VariableType	OperationType	Result of the interlock check could be data of an operation
Order	BaseData VariableType	OrderType	Result of the interlock check could be data of an order

3.4.7 ProcessParameterType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseProcessParameterType

This VariableType is created from the DataType

ProcessParameter

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseProcessParameterDataType

This type can be used to transfer values of a process parameter. It can be either a setpoint or an actual value. It can be used as a child in other structures or singularly. It contains the following members:

Table 3-33

Name	Type	Description
Id	Int32	A unique number assigned to the parameter
Name	String	The name of the parameter
Unit	EUInformation	A unique number assigned to the parameter OPC UA engineering unit information
Value	Number	Numeric value of the parameter (needs to be specified when instantiated)

3.4.8 DefectType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseDefectType

This VariableType is created from the DataType

Defect

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseDefectDataType

This type contains occurred defects including their status. It can be used as a child in other structures or singularly. It contains the following members:

Table 3-34

Name	Type	Description
Defect	DefectEnumType	Defect reason
Status	DefectStatusEnumType	Status of the defect

3.4.9 AlarmingType

urn:KitInformationmodelAlarming.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseAlarmingType

This VariableType is created from the DataType

Alarmig

urn:KitInformationmodelAlarming.Siemens.com
DataTypes > BaseDataType > Structure > BaseAlarmigDataType

This type contains the standard attributes of an automation system alarm diagnosing system alarms, warnings or informations (see subsection 2.4.3) occurring on the machine. For additional information and an application example of how to provide alarms over OPC UA server interface see https://support.industry.siemens.com/cs/ww/en/view/109748168.

Table 3-35

Name	Type	Description
AlarmText	String	Message text 254 characters
InfoText	String	Informational text
LC_ID	UInt16	Returns the Lcid's of the selected languages that are defined in the HW configuration.
ID_1	UInt16	Message ID is also displayed in the message editor
ID_2	UInt16	ID of the message runtime instance
PRIO	UInt16	Alarm priorities: 0 ... 16
ProducerID	UInt16	Identifier of the producer Possible values: 0: Program_Alarm 4: System diagnostics 5: Standard Motion Control 6: Security 7: SINUMERIK 8: GRAPH 9: ProDiag
TimeStamp	DateTime	Timestamp of generation
State	Byte	State of the alarm
AddText_1	String	Space for additional texts
AddText_2	String	Space for additional texts
AddText_3	String	Space for additional texts
AddText_4	String	Space for additional texts
AddText_5	String	Space for additional texts
AddText_6	String	Space for additional texts
AddText_7	String	Space for additional texts
AddText_8	String	Space for additional texts
AddText_9	String	Space for additional texts

3.5 Message content member types

3.5.1 OperationType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseOperationType

This VariableType is created from the DataType

Operation

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseOperationDataType

This type contains additional information about operations and which materials, equipment or process parameters are needed for it. It contains the following members:

Table 3-36

Name	Type	Description
OperationId	String	Id of the operation
OperationStatus	OrderStatusEnumType	Status of the operation
MaterialComponents	MaterialItemType[]	Array of to be used or produced material components
PlannedStartTime	DateTime	Planned start time of the operation
PlannedEndTime	DateTime	Planned end time of the operation
ProcessParameters	ProcessParameterType[]	Array of process parameters for the operation e.g. torque
EquipmentId	String	Id of the equipment or equipment group

3.5.2 MaterialItemType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseMaterialItemType

This VariableType is created from the DataType

MaterialItem

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseMaterialItemDataType

This type contains additional information about a used or produced material item. Could also be used for material changes, like quantities or process parameters. It contains the following members:

Table 3-37

Name	Type	Description
MaterialNo	String	Identifier of material type
BatchId	String	Id of the batch
SerialNo	String	Serial number of the material
MaterialQuantity	Number	Quantity of the material (could be float or int)
Unit	EUInformation	Unit (e.g., kg or liter)
MaterialSupplier	String	Name or Id of the material supplier
ProcessParameters	ProcessParameterType[]	Array of process parameters for the material e.g., torque

3.5.3 RecipeOperationType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseOperationType

This VariableType is created from the DataType

RecipeOperation

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseOperationDataType

This type contains additional information about recipe operations and which materials, equipment or process parameters are needed in this operation. It contains the following members:

Table 3-38

Name	Type	Description
OperationId	String	Id of the operation
MaterialComponents	MaterialItemType[]	Array of Material components to be used or consumed
ProcessParameters	ProcessParameterType[]	Array of process parameters for the material e.g., torque
EquipmentId	String	Id of the equipment or equipment group

3.5.4 RecipeMaterialItemType

urn:KitInformationmodel.Siemens.com
VariableTypes > BaseVariableType > BaseDataVariableType > BaseMaterialItemType

This VariableType is created from the DataType

RecipeMaterialItem

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Structure > BaseMaterialItemDataType

This type contains additional information about a used or consumed material item. It contains the following members:

Table 3-39

Name	Type	Description
MaterialNo	String	Identifier of material type
MaterialQuantity	Number	Quantity of the material (could be float or int)
Unit	EUInformation	Unit (e.g., kg or liter)
ProcessParameters	ProcessParameterType[]	Array of process parameters for the material e.g., torque

3.5.5 LocationTrackingEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the type of the item movement e.g. a material enters a station.

Table 3-40

Value	Description
Undefined_0	Undefined
Enter_1	Item is entering the station
Exit_2	Item is leaving the station

3.5.6 QualityStatusEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the status of the product quality.

Table 3-41

Value	Description
Undefined_0	Undefined
Good_1	Good
Bad_2	Bad

3.5.7 OrderStatusEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the status of the order.

Table 3-42

Value	Description
Undefined_0	Undefined
Edit_1	Can be modified
Pending_2	Operations and processes are assigned at a later point of time
ReadyForScheduling_3	This work order can be scheduled before it is released for production.
New_4	New
Active_5	Active
Hold_6	In case of problems in execution that need to be resolved
Queue_7	The work order assumes this status immediately after completion of one of its work order operations and prior to moving on the next work order operation of its sequence.
Pause_8	Pause
Split_9	work order has been split into smaller work orders
Rework_10	Rework
Scrap_11	The work order has been scrapped due to the presence of Non-Conformances of all parts that cannot be rectified.
Verified_12	Final product satisfies expectations

Value	Description
Complete_13	Complete
Abort_14	Abort

3.5.8 DefectEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the type of defect. It is highly depended on the customers/project requirements.

Table 3-43

Value	Description
Undefined_0	Undefined
Scratch_1	Scratch
BrokenElectronic_2	Broken electronic

3.5.9 DefectStatusEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the status of a specific defect.

Table 3-44

Value	Description
Undefined_0	Undefined
NewDefect_1	New defect
FixedDefect_2	Fixed defect
NotFixable_3	Cannot be fixed
RemainDefect_4	Remains defect

3.5.10 OperationStatusEnumType

urn:KitInformationmodel.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the status of an operation.

Table 3-45

Value	Description
Undefined_0	Undefined
Open_1	Open
Active_2	Active
Partial_3	Partial
FutureHold_4	Future hold
Complete_5	Complete
NotExecuted_6	Not executed

3.6 Opcenter message content objects

3.6.1 WorkOrderDownload

urn:MachineIntegrationProcessInstances.Siemens.com
Objects > WorkOrderDownload

This object is created from the ObjectType

WorkOrderDownload

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

It is used to download a created work order from Opcenter Execution Process in a generic way. It contains all the available attributes of a work order. If just some need to be sent, only those are set. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Process to fulfill the **inbound** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-46

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
equipment	String	Input. Equipment where order needs to be executed
id	String	Input. WorkOrderIdentifier from MES
isFullDownload	Boolean	Input. Full structure download
material	String	Input. Id of Material that should be produced
materialName	String	Input. Name of material that should be produced
materialRevision	String	Input. Revision of Material that should be produced
name	String	Input. WorkOrderName from MES
plannedEndTime	DateTime	Input. Time/Date when Order should be finished
plannedStartTime	DateTime	Input. Time/Date when Order should be started
quantity	Float	Input. Quantity that should be produced
uom	String	Input. UoM of Material that should be produced
workMaster	String	Input. Identifier of Recipe/WorkMaster with which it should be produced
workMasterRevision	String	Input. Revision of Recipe/WorkMaster with which it should be produced
numberOf ParametersBool	Int16	Input. Contains the number of Boolean Parameters transfered in the array
numberOf ParametersDateTime	Int16	Input. Contains the number of DateTime Parameters transfered in the array
numberOf ParametersLInt	Int16	Input. Contains the number of LInt Parameters transfered in the array
numberOf ParametersLReal	Int16	Input. Contains the number of LReal Parameters transfered in the array

Name	Type	Description
numberOf ParametersString	Int16	Input. Contains the number of String Parameters transfered in the array
numberOfOperations	Int16	Input. Contains the number of workOrderOperations transfered in the array
parametersBool	ParameterBool[]	Input. Contains the Boolean Parameters
parametersDateTime	ParameterDateTime[]	Input. Contains the DateTime Parameters
parametersLInt	ParameterLInt[]	Input. Contains the LInt Parameters
parametersLReal	ParameterLReal[]	Input. Contains the LReal Parameters
parametersString	ParameterString[]	Input. Contains the String Parameters
workOrderOperations	WorkOrderOperation[]	Input. Contains the workOrderOperations

3.6.2 ChangeStatusRequest

urn:MachineIntegrationProcessInstances.Siemens.com
Objects > ChangeStatusRequest

This object is created from the ObjectType

StatusChange

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseStatusChange

It requests a status change from the shopfloor. As only shopfloor is able to influence the real status, it can be successful or not. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Process to fulfill the **inbound** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-47

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
statusTransition	String	Input. Contains the requested status transition
timestamp	DateTime	Input. Timestamp
workOrder	String	Input. Identifier of the WorkOrder that should change the status
workOrderOperation	String	Input. Identifier of the WorkOrderOperation that should change the status

3.6.3 ChangeStatusNotification

urn:MachineIntegrationProcessInstances.Siemens.com
Objects > ChangeStatusNotification

This object is created from the ObjectType

StatusChange

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseStatusChange

It notifies Opcenter that a status change happened. This can be either requested or autonomously done. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Process to fulfill the **outbound** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-48

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
statusTransition	String	Input. Contains the occurred status transition
timestamp	DateTime	Input. Timestamp
workOrder	String	Input. Identifier of the work order that changed the status
workOrderOperation	String	Input. Identifier of the work order operation that changed the status

3.6.4 ExecuteMaterialTransaction

urn:MachineIntegrationProcessInstances.Siemens.com
Objects > ExecuteMaterialTransaction

This object is created from the ObjectType

ExecuteMaterialTransaction

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteMaterialTransaction

It reports a material transaction from shopfloor in Opcenter Execution Process. It can be a movement, consumption, production or creation, depending on the filled data. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Process to fulfill the **outbound** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-49

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
action	ExecuteMaterialTransactionEnumType	Input. Action that should be executed
actualMaterialSequence	Int16	Input. The sequence number that shows in which sequence the material was consumed
Destination Equipment	String	Input. Equipment from MES where the unit is moved or consumed to
Destination Quantity	Float	Input. Destination quantity of material that is moved or consumed
lot	String	Input. The known lot identifier of the unit moved or consumed
material	String	Input. The material that the moved or consumed unit is of
materialRevision	String	Input. The material revision that the moved or consumed unit is of
sourceEquipment	String	Input. Equipment from Opcenter where the unit is moved or consumed from
sourceQuantity	Float	Input. Source quantity of material that is moved or consumed
timestamp	DateTime	Input. Time at which the movement or consumption was made
workOrder	String	Input. Work order in which context the movement or consumption is done
workOrderOperation	String	Input. Work order operation in which context the movement or consumption is done

3.6.5 ExecuteMaterialTransactionEnumType

urn:MachineIntegrationProcess.Siemens.com
DataTypes > BaseDataType > Enumeration

This type identifies the action of a MaterialTransaction.

Table 3-50

Value	Description
MaterialMovement_0	Move an MTU from source to destination
MaterialConsumption_1	Moves Material and additionally record Work Order Operation Actual Material as Material Consumption
MaterialProduction_2	Moves Material and additionally record Work Order Operation Actual Material as Material Production
MaterialProductionAndConsumption_3	Moves Material and additionally record Work Order Operation Actual Material as Material Production and Consumption
ExternalMaterialIncoming_4	Record a new material that arrives in the plant creating the lot and a Material Tracking Unit with the same quantity from the lot.

3.6.6 ExecuteSaveSequence

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > ExecuteSaveSequence

This object is created from the ObjectType

ExecuteSaveSequence

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteSaveSequence

Is called in the first station of a production line. Calling this request saves the scanned serial number in sequence for each of the following configured equipments, so it can be consumed later by each of the equipments in the saved order by calling RequestNext. This way, knowledge of the productionId is not necessary for the machines in a production line, as long as items are processed sequentially. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the **request** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-51

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify Work Order. Can be the MTU Code, the MTU NId or the WO NId
returnValue	Int16	Output. Return Value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.7 RequestNext

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > RequestNext

This object is created from the ObjectType

RequestNext

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseRequestNext

Calling this request returns the next item to process. It can either be the one previously saved (by calling ExecuteSaveSequence) or a different one defined by StartDate or other attributes. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the **request** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-52

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
Material TrackingUnit	String	Output. Code of the next material tracking unit (Serial number or batch number) to be processed.
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.8 RequestOrderInfo

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > RequestOrderInfo

This object is created from the ObjectType

RequestOrderInfo

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseRequestOrderInfo

Calling this request returns additional order information about the requested productionId. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the **request** handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-53

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify work order. This can be the MTU Code, the MTU NId or the WO NId
workOrder	String	Output. Natural identifier of final the work order.
quantity	Float	Output. Initial quantity of the work order.
material	String	Output. Natural identifier of final material of the work order.
Material TrackingUnit	String	Output. Natural identifier of the first material tracking unit (Serial number or batch no) associated to work order.
ERPOrder	String	Output. ERP Order value of the work order.
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.9 ExecuteWorkOrderOperation

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > ExecuteWorkOrderOperation

This object is created from the ObjectType

ExecuteWorkOrderOperation

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteWorkOrderOperation

Calling this request executes a work order operation action (e.g. start or stop) on the requested operation. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the request handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-54

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify the work order.
action	String	Input. Defines the action to do on the work order operation
batchQuantity	Int16	Input. Defines the quantity of the batch that this action is done on in case it is not serialized
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.10 ExecuteConsumeMaterials

urn:MachineIntegrationDiscrete.Siemens.com
Objects > ExecuteWorkOrderOperation

This object is created from the ObjectType

ExecuteMaterialConsumption

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseMaterialConsumption

Calling this request executes a material consumption of the materials from the input to the component defined by productionId. They will be registered as consumed in the Operation which is being executed on the current station. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the request handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-55

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify the material tracking unit or work order. Can be the MTU Code, the MTU NId or the WO NId
logicalPosition	String[3]	Input. Logical position of the material tracking unit consumed by the material tracking unit or work order.
material	String[3]	Input. Material identifier of the material tracking unit consumed by the material tracking unit or work order.
materialTracking UnitCode	String[3]	Input. Code of the material tracking unit consumed by the material tracking unit or work order.
quantity	Float[3]	Input. Quantity of the material tracking unit consumed by the material tracking unit or work order.
numMaterials	Int16	Input. Number of materials that are valid inside of the arrays and that are consumed by the material tracking unit or work order.
Material ConsumptionStatus	Int16[3]	Output. Return values for the single materials inside of the arrays. Represents if a material has been consumed or not. If respective value is 0 then material is consumed else not.
returnValue	Int16	Output. Return Value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.11 RequestMaterialConsumption

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > RequestMaterialConsumption

This object is created from the ObjectType

RequestMaterialConsumption

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseMaterialConsumption

Calling this requests the material requirements to be consumed from the component defined by productionId. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the request handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-56

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify the material tracking unit or work order. Can be the MTU Code, the MTU NId or the WO NId
logicalPosition	String[3]	Output. Logical position of the material tracking unit consumed by the material tracking unit or work order.
material	String[3]	Output. Material identifier of the material tracking unit consumed by the material tracking unit or work order.

Name	Type	Description
quantity	Float[3]	Output. Quantity of the material tracking unit consumed by the material tracking unit or work order.
numMaterials	Int16	Output. Number of materials that are valid inside of the arrays and that were requested to be consumed by the material tracking unit or work order.
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.12 ExecuteQualityInspection

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > ExecuteQualityInspection

This object is created from the ObjectType

ExecuteQualityInspection

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteQualityInspection

Calling this request triggers a Quality Inspection on Opcenter side. The object can be directly mapped to the Automation Node Instance of Opcenter Execution Discrete to fulfill the request handshake implementation and execute the out-of-the-box functionality.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-57

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify the material tracking unit or work order. Can be the MTU Code, the MTU NId or the WO NId
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.13 ExecuteWorkInstructionDataCollection

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > ExecuteWorkInstructionDataCollection

This object is created from the ObjectType

ExecuteWorkInstructionDataCollection

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteWorkInstructionDataCollection

Calling this request triggers a Data Collection on opcenter side.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-58

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
productionId	String	Input. Contains the value used to identify the material tracking unit or work order. Can be the MTU Code, the MTU NId or the WO NId
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.14 ExecuteTrackItemLocation

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > ExecuteTrackItemLocation

This object is created from the ObjectType

ExecuteItemLocationTracking

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteTrackItemLocation

Calling this request tracks several items inside several equipments.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-59

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
numItems	Int16	Input. Describes how many items of the following arrays are valid
itemIds	String[10]	Input. IDs if the tracked item
equipmentIds	String[10]	Input. EquipmentId from Opcenter where the item is currently located
actions	LocationTracking Action[10]	Input. Action that was performed lastly (start, warning, exiting, exit)
errorCodes	Int16[10]	Output. Returns the error codes of the items after tracking transaction
milestones	String[10]	Output. Returns the milestones that were reached
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.15 ExecuteTrackItemLocationSingle

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > ExecuteTrackItemLocationSingle

This object is created from the ObjectType

ExecuteItemLocationTrackingSingle

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseExecuteTrackItemLocation

Calling this request tracks one item inside an equipment.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-60

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
itemId	String	Input. ID if the tracked item
equipmentId	String	Input. EquipmentId from Opcenter where the item is currently located
action	Location TrackingAction	Input. Action that was performed lastly (start, warning, exiting, exit)
errorCode	Int16	Output. Returns the error code of the items after tracking transaction
milestone	String	Output. Returns the milestone that was reached
returnValue	Int16	Output. Return value is used to identify whether a request is successfully processed or not. In case of success its value is set to 0. In case of failure, its value will represent error code.

3.6.16 LocationTrackingAction

urn:MachineIntegrationDiscrete.Siemens.com
DataTypes > BaseDataType > Enumeration > LocationTrackingAction

This type identifies the type of a Location Tracking action.

Table 3-61

Value	Description
Invalid_0	Invalid
Start_1	Item enters the equipment
Warning_2	Item is at 50% of the equipment (of as defined)
Exiting_3	Item is at 70% of the equipment (of as defined)
Exit_4	Item exited the equipment

3.6.17 DownloadMap

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > DownloadMap

This object is created from the ObjectType

BaseObjectType

Object to send an equipment map to the machine as described in the provided Opcenter application example.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-62

Name	Type	Description
sequenceNo	UInt32	Input. Initiating trigger of the tag-based handshake mechanism
acknowledgeNo	UInt32	Output. Acknowledging trigger of the tag-based handshake mechanism
msgInterface	typeDownload MapEnvelope	Input. Contains the envelope to transfer an equipment map to the machine as implemented in the provided application example.

3.6.18 RuntimeLocationMaps

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > RuntimeLocationMaps

This object is created from the ObjectType

BaseObjectType

This object contains an example of runtime maps described in the provided Opcenter application example.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-63

Name	Type	Description
itemLocations	typeRuntime LocationMap	Contains the overall locations and which items are currently processed inside
itemLocationsLeft	typeRuntime LocationMap	Contains the locations on the left side of the line and which items are currently processed inside
itemLocationsRight	typeRuntime LocationMap	Contains the locations on the right side of the line and which items are currently processed inside

3.6.19 typeRuntimeLocationMap

urn:MachineIntegrationDiscrete.Siemens.com
DataTypes > BaseDataType > Structure > typeRuntimeLocationMap

This type describes a map where the items at runtime can be displayed.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-64

Name	Type	Description
numberOfLocations	Int16	describes, how far the following arrays are filled with valid equipments
equipmentIds	String[6]	Contains the equipmentId from Opcenter
items	String[6]	Contains the Item that is processed in the equipment, if so
actions	Int16[6]	Contains the status of the item in the equipment (Started=1, warning=2, exiting=3)
itemIndex	Int16[6]	Reference to the Item array from the configured Item map (see DownloadMap)

3.6.20 Communication (discrete)

urn:MachineIntegrationDiscreteInstances.Siemens.com
Objects > Communication

This object is created from the ObjectType

CommunicationMID

urn:MachineIntegrationDiscrete.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > CommunicationObjectType

This object provides some communication related mechanisms.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Discrete License**, you can find further information here: https://support.sw.siemens.com/en-US/product/209927011/knowledge-base/PL8697155

Table 3-65

Name	Type	Description
ClearAllMessages	Boolean	Deletes all messages in all buffers and resets sequence and acknowledge numbers and retries
HeartbeatSet	Boolean	Should be set every heartbeat interval so that machine can evaluate the connection to the client
RestartRetryExecItem LocationTracking	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryExecItem LocationTrackingSingle	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryExecMaterial ConsumptionHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryExecQuality InspectionHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryExec SaveSeqHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryExecWI CollectionHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract

Name	Type	Description
RestartRetryExec WOOHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryRequest MaterialConsumptionHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryReques tNextHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryRequest OrderInfoHsk	Boolean	In case Restart expired, this trigger restarts the retry of this contract

3.6.21 Communication (process)

urn:MachineIntegrationProcessInstances.Siemens.com
Objects > Communication

This object is created from the ObjectType

CommunicationMIP

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > CommunicationObjectType

This object provides some communication related mechanisms.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-66

Name	Type	Description
ClearAllMessages	Boolean	Deletes all messages in all buffers and resets sequence and acknowledge numbers and retries
HeartbeatSet	Boolean	Should be set every heartbeat interval so that machine can evaluate the connection to the client
RestartRetry MaterialTransaction	Boolean	In case Restart expired, this trigger restarts the retry of this contract
RestartRetryStatus ChangeNotification	Boolean	In case Restart expired, this trigger restarts the retry of this contract

3.6.22 WorkOrderOperation

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process work order operation. It describes an operation done within a work order.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-67

Name	Type	Description
equipment	String	Equipment where operation should be executed
id	String	Identifier of work order operation that should be executed
operation	String	Operation definition that defines what to execute in this operation
operationRevision	String	Operation revision definition that defines what to execute in this operation
name	String	Name of work order operation that should be executed
plannedEndTime	DateTime	Time/Date when operation should be finished
plannedStartTime	DateTime	Time/Date when operation should be started
workOrderOperation Sequence	Int16	Sequence of operation, how it should be executed
numberOfParameters Bool	Int16	Contains the number of Boolean Parameters transfered in the array
numberOfParameters DateTime	Int16	Contains the number of DateTime Parameters transfered in the array
numberOfParameters LInt	Int16	Contains the number of LInt Parameters transfered in the array
numberOfParameters LReal	Int16	Contains the number of LReal Parameters transfered in the array
numberOfParameters String	Int16	Contains the number of String Parameters transfered in the array
numberOfEquipments	Int16	Contains the number of Equipments transfered in the array
numberOfMaterials	Int16	Contains the number of Materials transfered in the array
parametersBool	ParameterBool[]	Contains the Boolean Parameters
parametersDateTime	ParameterDateTime[]	Contains the DateTime Parameters
parametersLInt	ParameterLInt[]	Contains the LInt Parameters
parametersLReal	ParameterLReal[]	Contains the LReal Parameters
parametersString	ParameterString[]	Contains the String Parameters
workOrderOperation Equipments	WorkOrderOperationEquipment[]	Contains the workOrderOperationEquipments
workOrderOperation Materials	WorkOrderOperationMaterial[]	Contains the workOrderOperationMaterials

3.6.23 WorkOrderOperationEquipment

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process work order operation equipment. It describes an equipment to be used within a work order operation.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-68

Name	Type	Description
Equipment Sequence	Int16	Sequence in which the equipment needs to be used
id	String	Identifier of the Equipment
name	String	Name of the Equipment

3.6.24 WorkOrderOperationMaterial

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process work order operation material. It describes a material to be consumed or produced within a work order operation.

For further information please contact SIMATIC Systems Support (FA S SUP). As a customer with **Opcenter Execution Process License**, you can find further information here: https://support.sw.siemens.com/en-US/product/279851068/knowledge-base/PL8663204

Table 3-69

Name	Type	Description
materialSequence	Int16	Sequence of the to be consumed or produced material
id	String	Identifier of the material to be consumed or produced in this operation
name	String	Name of the material to be consumed or produced in this operation
equipment	String	Equipment where material needs to be consumed or produced
direction	String	Input or output direction
lot	String	Known Lot of the material to be consumed or produced
quantity	Float	Quantity of the to be consumed or produced material
revision	String	Revision of the material to be consumed or produced in this operation
uom	String	Unit of measurement of the to be consumed or produced material
usage	String	Usage of material

3.6.25 ParameterBool

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process boolean parameter. It describes a parameter of type boolean which can be an attribute of a work order operation or an order. It contains the following members:

Table 3-70

Name	Type	Description
equipment	String	Equipment of the parameter
id	String	Identifier of the parameter
name	String	Name of the parameter
targetValue	Boolean	Value of the parameter

3.6.26 ParameterDateTime

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process date or time parameter. It describes a parameter of type DateTime which can be an attribute of a work order operation or an order. It contains the following members:

Table 3-71

Name	Type	Description
equipment	String	Equipment of the parameter
id	String	Identifier of the parameter
name	String	Name of the parameter
targetValue	DateTime	Value of the parameter

3.6.27 ParameterLInt

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process integer parameter. It describes a parameter of type LInt which can be an attribute of a work order operation or an order. It contains the following members:

Table 3-72

Name	Type	Description
equipment	String	Equipment of the parameter
id	String	Identifier of the parameter
name	String	Name of the parameter
targetValue	Int64	Value of the parameter

3.6.28 ParameterLReal

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process real parameter. It describes a parameter of type LReal which can be an attribute of a work order operation or an order. It contains the following members:

Table 3-73

Name	Type	Description
equipment	String	Equipment of the parameter
id	String	Identifier of the parameter
name	String	Name of the parameter
targetValue	Double	Value of the parameter

3.6.29 ParameterString

urn:MachineIntegrationProcess.Siemens.com
ObjectTypes > BaseObjectType > BaseOpcenterObjectType > BaseWorkOrderDownload

This type defines an Opcenter Execution Process string parameter. It describes a parameter of type String which can be an attribute of a work order operation or an order. It contains the following members:

Table 3-74

Name	Type	Description
equipment	String	Equipment of the parameter
id	String	Identifier of the parameter
name	String	Name of the parameter
targetValue	String	Value of the parameter

3.7 Others

3.7.1 CommunicationObjectType

urn:BaseInformationmodelkit.Siemens.com
ObjectTypes > BaseObjectType > BaseCommunicationObjectType

Implements communication interfaces for some general functionalities within IT/OT communication. It contains the following (optional) members:

Table 3-75

Name	Type	Description
ClearAllMessages	Boolean	Can be set from the PLC or from outside and resets all message handshakes and deletes all messages from the message buffers without stopping the whole machine
HeartbeatGet	Boolean	Needs to be set and reset by the PLC once per predefined cycle and shows the upper-level system whether the connection to the server is still available
HeartbeatSet	Boolean	Needs to be set by the upper-level system once per predefined cycle (is automatically reset by the plc after a while) and shows the PLC whether the connection to the client is still available.
RestartRetryExec SaveSeqHsk	Boolean	Can be set from the PLC or from outside and restarts the retry of this certain message handshake in case it is expired.
RestartRetryExec WOOHsk	Boolean	Can be set from the PLC or from outside and restarts the retry of this certain message handshake in case it is expired.
RestartRetry RequestNext	Boolean	Can be set from the PLC or from outside and restarts the retry of this certain message handshake in case it is expired.
RestartRetry RequestOrderInfo	Boolean	Can be set from the PLC or from outside and restarts the retry of this certain message handshake in case it is expired.
RestartRetry MaterialTransaction	Boolean	Can be set from the PLC or from outside and restarts the retry of this certain message handshake in case it is expired.
RestartRetry StatusChange Notification	Boolean	Can be set from the PLC or from outside and restarts the retry of this certain message handshake in case it is expired.

4 How to use the Information Model with SiOME

The following instructions are based on these versions:

Table 4-1

Product	Version
TIA Portal	17
SiOME	2.5.10

If the available versions are different, check the corresponding documentation for any changes.

In the following a reasonable Information Model will be instantiated from the provided types.

4.1 Introduction

The information model you are creating here, will be a standard OPC UA Nodeset in the common .xml format. You are using DataTypes, ObjectTypes and VariableTypes to instantiate an information model and map it to a TIA Portal project. An information model can be seen as the description of the interface to one machine. It does usually not contain a cluster of machines if it is not a header PLC and speaks for underlying systems. If it is created in a standardized way for many machines (of a kind) it can just be loaded to all of these machines.

Brownfield:

As already existing automation systems are sometimes not supposed (or needed) to be touched, the creation of a standard information model can be done by mapping the model to already existing tags in the TIA Portal project and loading the interface to the OPC UA server running on this machine. Missing use cases of message transfer still might need some work to be done on the machine, as messages are usually not sent anyway, even if not needed.

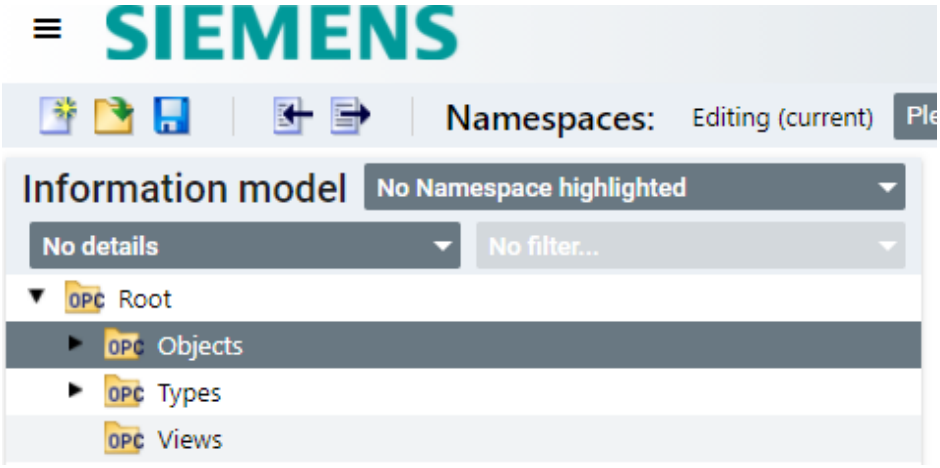
Greenfield:

The interface can either be directly imported to you TIA Portal project and then be implemented after this specification or also just be mapped to an existing specification or even just be mapped and handed out to be implemented as a specification.

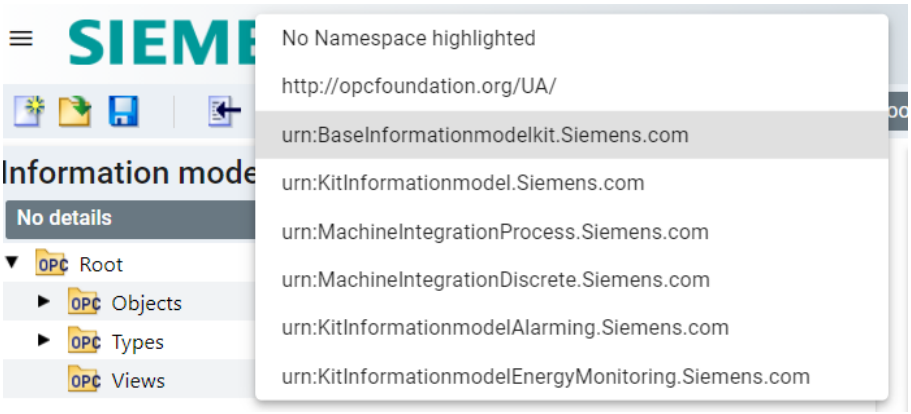
4.2 Basic setup

To start creating your own standardized information model, open SiOME .

- Import the provided nodeset-XML files . Always start with the BaseInformationModel. Then you can add others according subchapter Node sets and namespaces
- Highlight the nodes of the base information model kit by clicking the dropdown list next to "Information model"

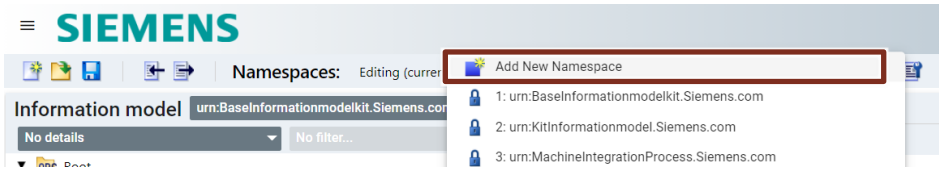


- Then select the namespace "urn:BaseInformationmodelkit.Siemens.com"

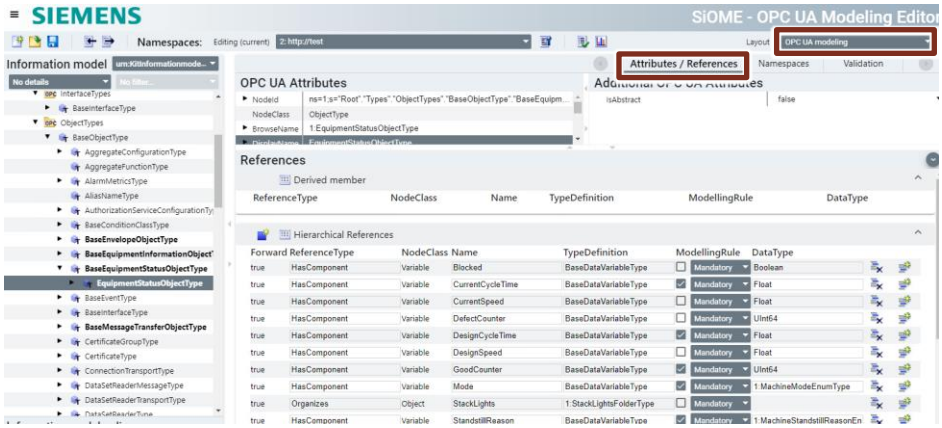


- If a namespace imported by one of the files adds Nodes to the address space, you will find
  - The DataTypes within
    Root > Types > DataTypes > BaseDataType > Structure
  - TheInterfaceTypes within
    Root > Types > InterfaceTypes > BaseInterfaceType
    or
    Root > Types > ObjectTypes > BaseObjectType > BaseInterfaceType
  - The ObjectTypes within
    Root > Types > BaseObjectType
  - The VariableTypes within
    Types > VariableTypes > BaseVariableType > BaseDataVariableType

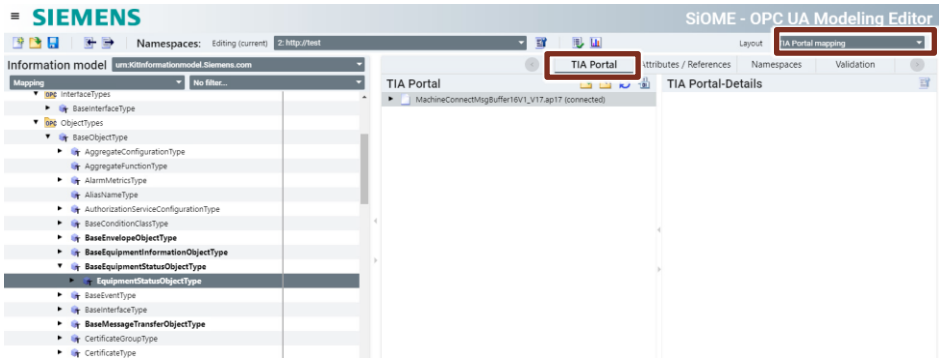
- To create your own Information Model, first create an own namespace by clicking on the editing dropdown menu and choose "Add New Namespace". The namespace uri can be anything.



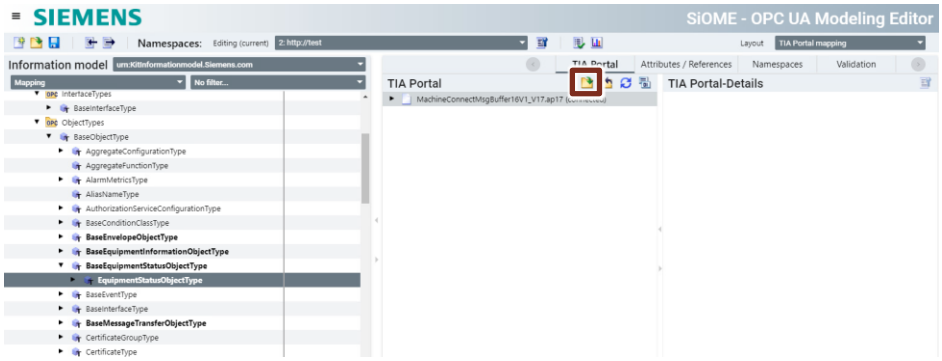
- Choose the Layout "OPC UA modelling" (Attributes/References Tab) to see the references of an ObjectType or VariableType or Object.



- Choose the Layout "TIA Portal mapping" (TIA Portal Tab) to connect and map to a TIA Portal Project



- To connect it to a TIA Portal Project, click on "Open TIA Project" on the top of the middle pane, select your project from the displayed (running in the background) projects and attach it.

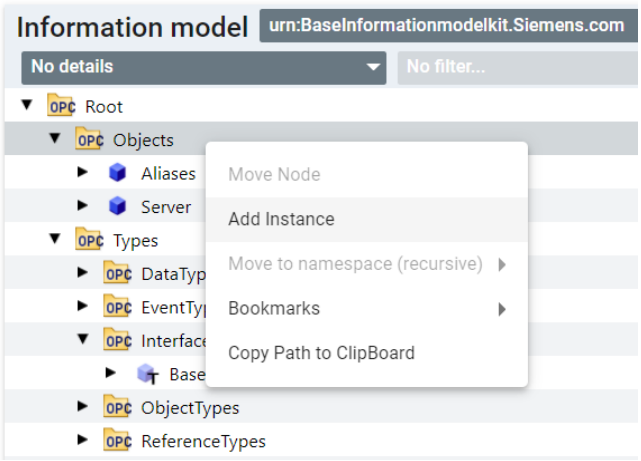


- Before refreshing or opening, save and compile your project. After opening the first time (or loading an interface to the TIA Portal project), check whether TIA Portal needs your action.

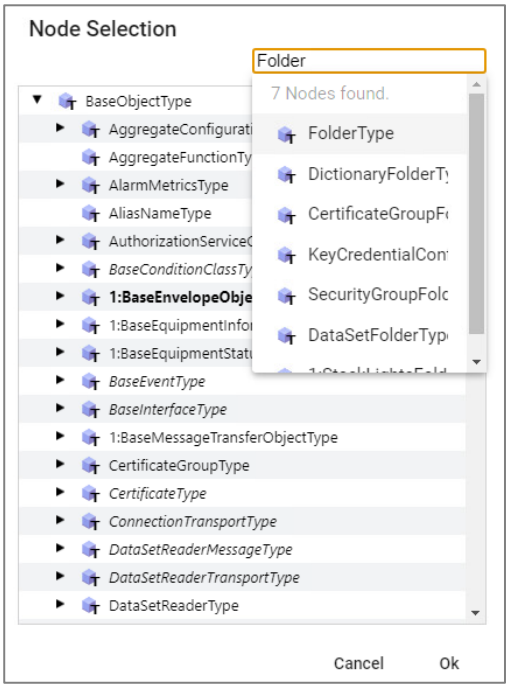
4.3 Create continuous machine data objects

- To create continuous machine data objects (see also sections 1.2.2, 3.1 and 3.2), you need to instantiate the corresponding objects. For this, import the namespace "urn:KitInformationmodel.Siemens.com".

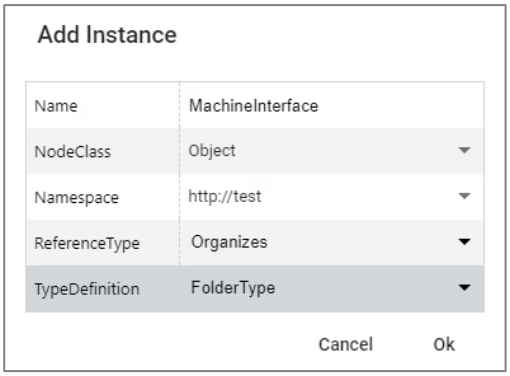
- In Root > Objects, Add a new Instance



- As continuous machine data is often already existing on the machine without additional implementation effort, you could create a dedicated folder "MachineInterface" which will later contain tags to be mapped to existing PLC-tags. As TypeDefinition of this Object search for the OPC UA ObjectType "FolderType"



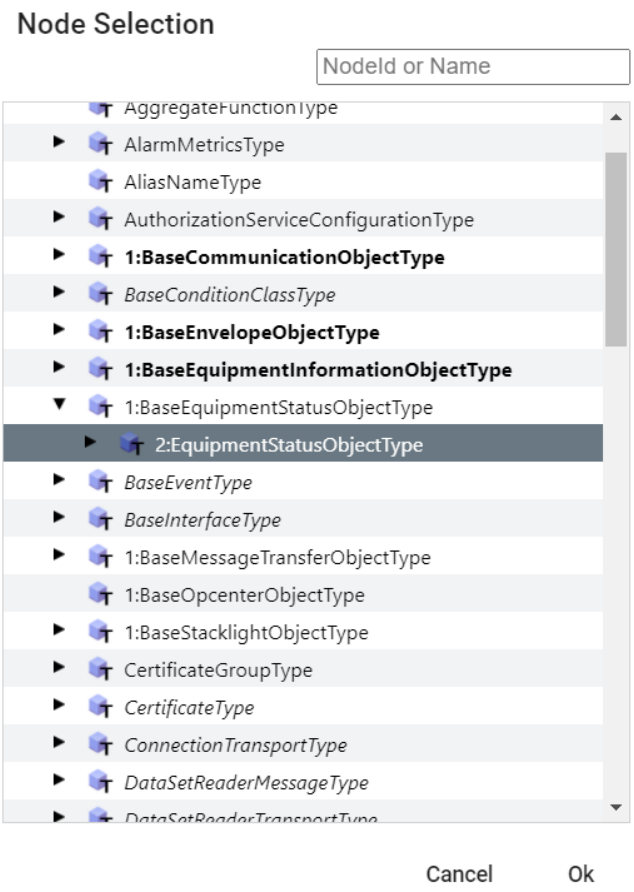
- Add the Instance

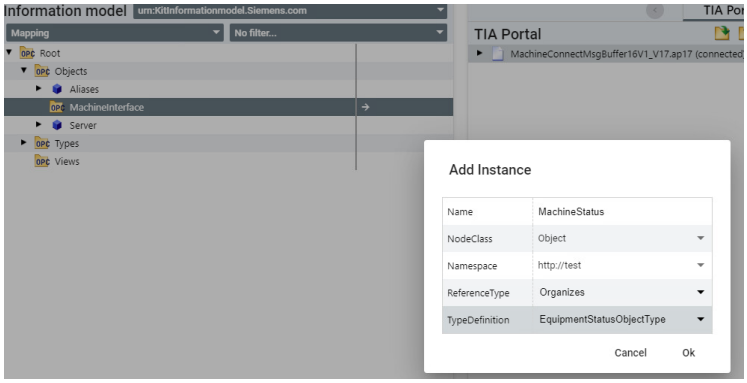


- Create another Instance of NodeClass Object within this FolderType Object. This time, create an Object of the provided ObjectType EquipmentStatusObjectType.

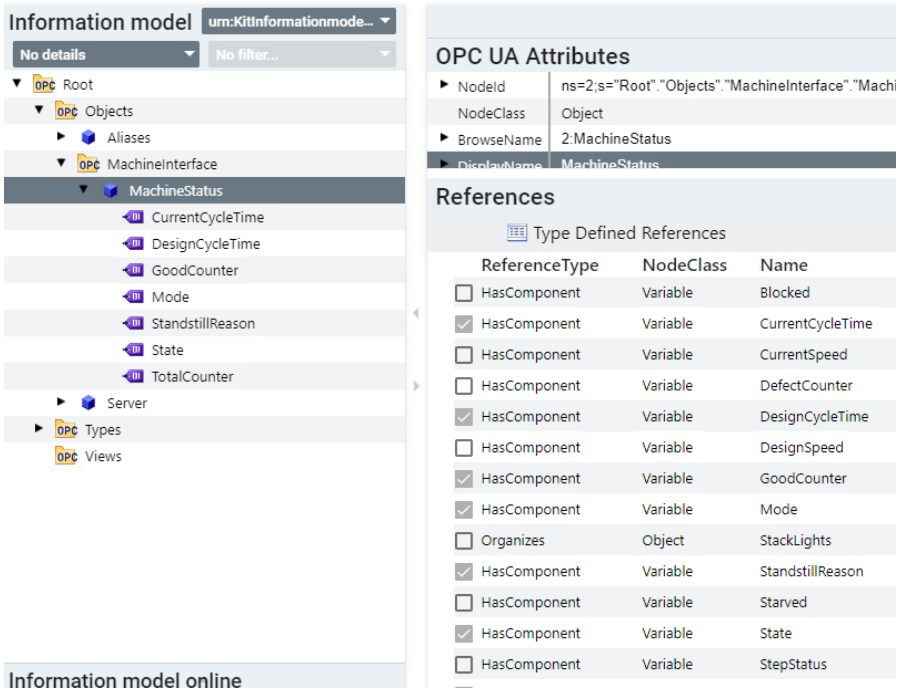
NOTE

When choosing a Data-, Variable- or ObjectType, you always need to expand the (abstract) parent ObjectType, DataType or VariableType to reach the concrete one. Otherwise, if you choose an abstract type, your instance won't have any content. The abstract types are usually named starting with "Base…".

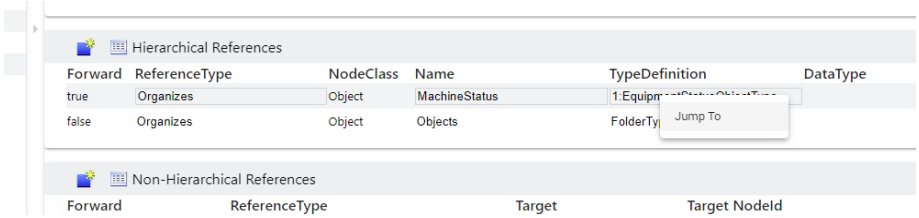




- After creating, you can see, that some children of your new Object are not shown.

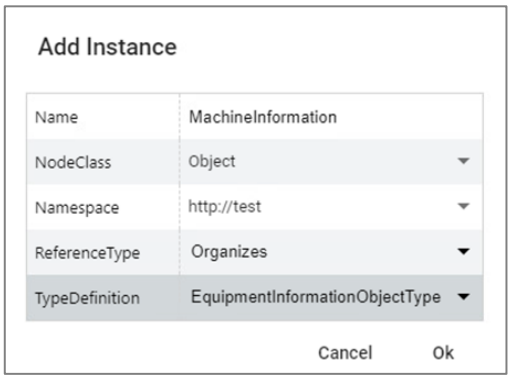


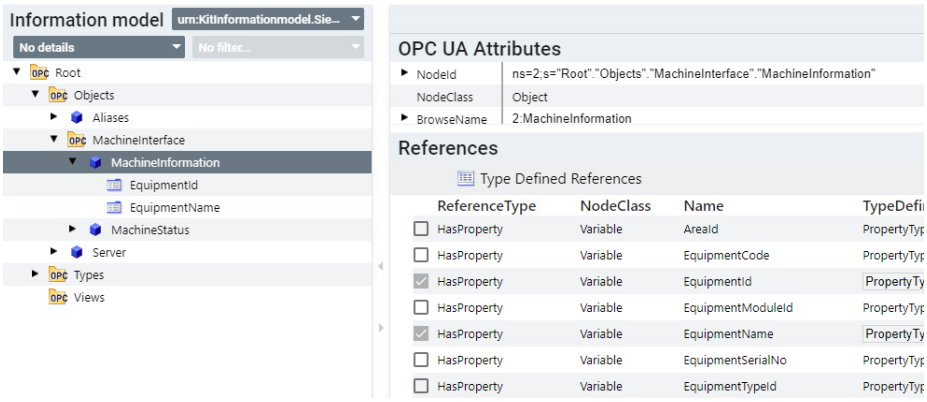
- You can select, which children are added to the server interface by selecting the checkboxes on the right side. Some of them are mandatory. If you nevertheless still do not want to have them in your information model, go to the DataType "EquipmentStatusObjectType" to change this (see also section Create own types 4.6). You can get there if you rightclick on the type and select "Jump To".



NOTE To change a type within a different namespace, you need to set the editing namespace to the one you want to change (e.g., EquipmentStatusObjectType: unlock namespace 1). See chapter 3 for namespaces of the types

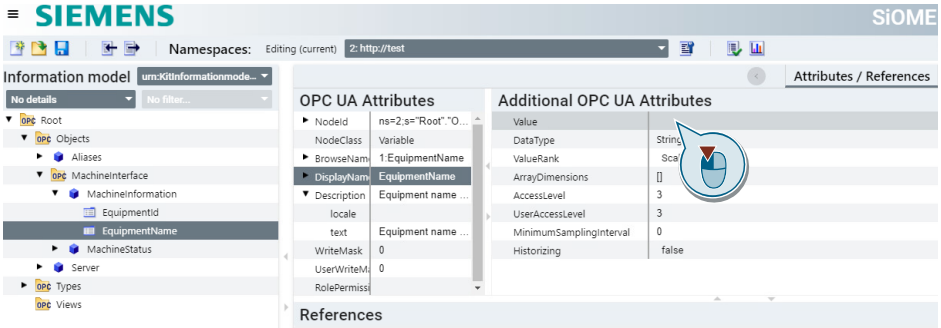
- If desired, create an Object "MachineInformation" as previously described for the MachineStatus. As TypeDefinition use the "EquipmentInformationObjectType"

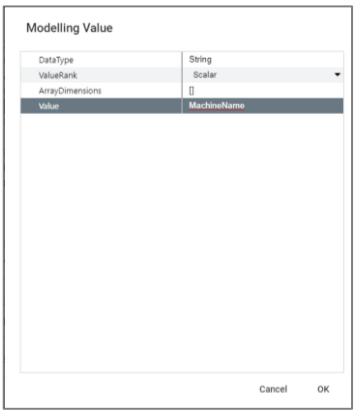




The HasComponent reference type is used for variables that change dynamically (counters, states, …). The HasProperty reference type is used for variables that are usually constant (name, id, …)

- Variables do not need to be mapped to real tags within the machine code. They can be statically defined in the OPC UA server. To set the value of a Property or a constant Component to a fixed value just provided within the OPC UA server interface stored on the PLC, click on the corresponding Node and change "Value"





4.4 Create message structures

- To create message transmission objects, you might want to create a new FolderObject that represents another DB inside the TIA Portal Project. Separating them from the continuous machine data objects, helps you importing that part to TIA Portal and you can then enrich the interface data with communication data like the message buffer and the managing buffer handler data, which you could do in the same DB.

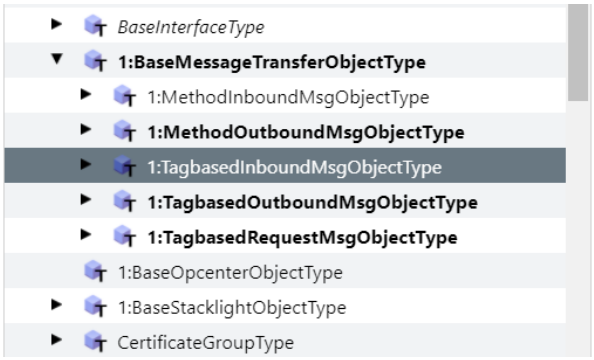
NOTE

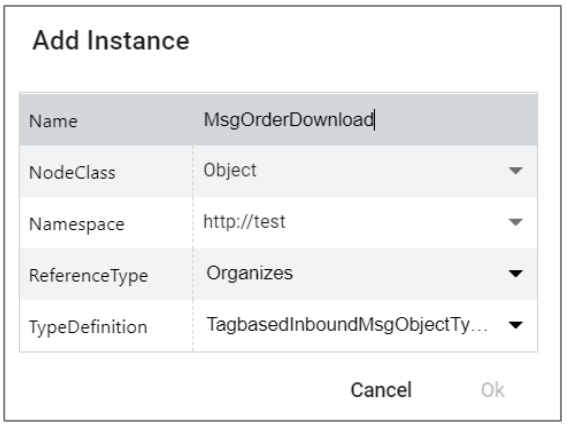
The maximum nesting depth of a TIA Portal data structure is 9. For some of the provided DataTypes, given that you want to use them full-blown, it might be necessary to create the message objects directly under objects without an intermediate folder. This applies especially if you need to import them into TIA Portal.

- For this, add a new Instance of FolderType in Root > Objects called e.g. "MachineInterfaceMessages" just like before.
- Within this folder object, add another instance of the message transfer type under BaseMessageTransferObjectType you need. You can decide between tag-based communication and method-based communication and whether the PLC sends or receives the message or wants to send a request message that returns a response immediately.

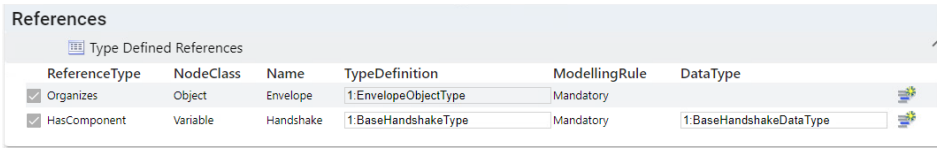
4.4.1 Tag-based communication

- For tag-based communication choose the ObjectType TagbasedInboundMsgObjectType or TagbasedOutboundMsgObjectType

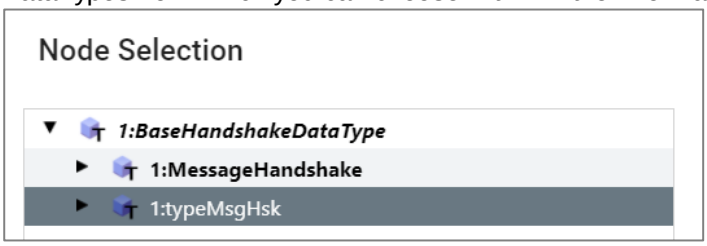




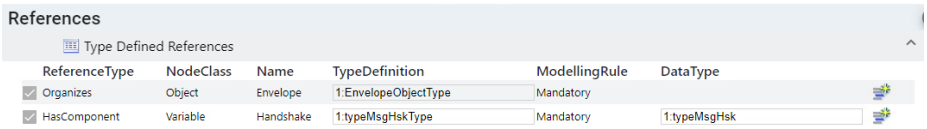
It will come with a Handshake variable to do the handshake on, and an Envelope Object, that contains the message. The Envelope is already a non-abstract ObjectType that contains header and content, the handshake DataType is on the other hand still abstract and needs to be specified.



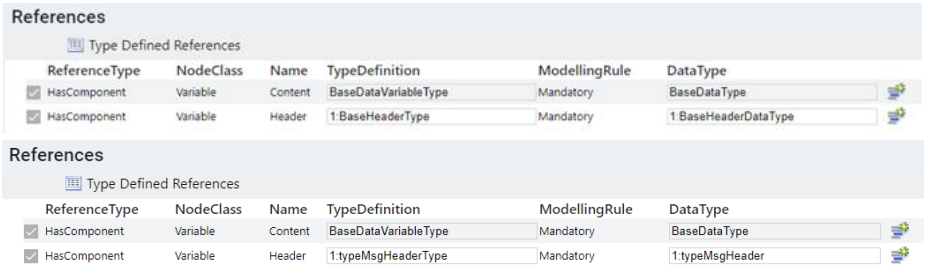
- To specify the handshake DataType, click in BaseHandshakeDataType. There are two DataTypes from which you can choose within in the Information Model Kit.



- The type MessageHandshakeType is a handshake structure definition in conformity with standard OPC UA Companion specification style, useful for mapping. The type "typeMsgHsk" is the direct DataType used in the "MachineConnect" application example to implement a handshake-buffer-mechanism. Using this type, the DBs can be directly imported into TIA Portal and be reused to implement message buffers accordingly. It is named after the Siemens style guide for TIA Portal Projects. (see https://support.industry.siemens.com/cs/ww/en/view/81318674)
- As you can also see, you can only choose between these two types as the Variable is defined as the BaseHandshakeDataType and therefore restricts the choice to the children types.
- Repeat this step for the BaseHandshakeType.



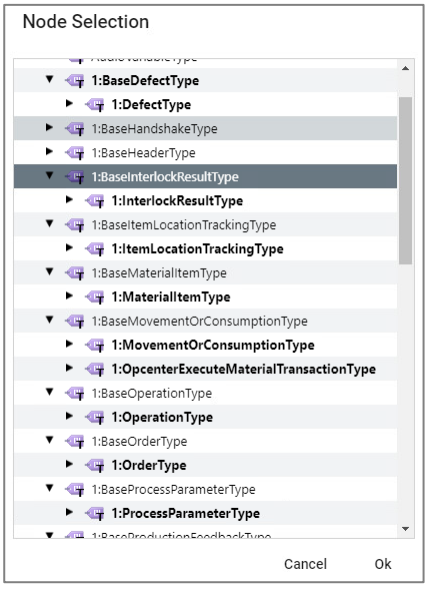
- Go to the Envelope Object. You need again to specify the types of the header variable, where the exact same options are available as in the handshake structure.

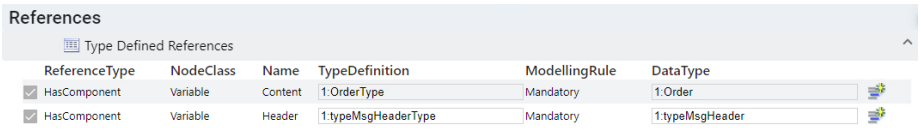


NOTE

If you want to set the types for header and handshake structures generally and not only for each single instance you can specify it in the InterfaceType of the required message object structure before creating the objects. All the instances will then inherit this choice and you don't need to specify it anymore.

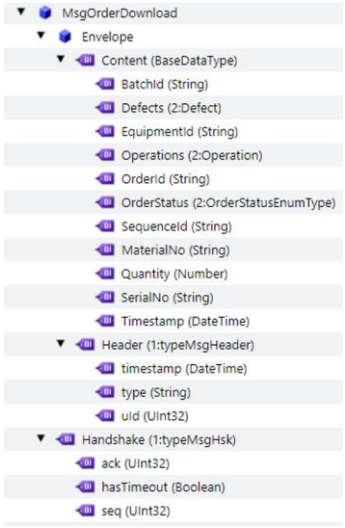
- Now the structure is built up, but the message still needs its payload. For this, choose the right types for the variable Content out of all possible BaseDataTypes and BaseDataVariableTypes. This can be basically everything, either one of the suggested use-case data types, own types or just elements of the existing types like ProcessParameters, Operations or Defects.





- Choose OrderType and Order for the OrderDownload message structure as the "Content" variable, as it is designed to contain all the order information that can be downloaded.

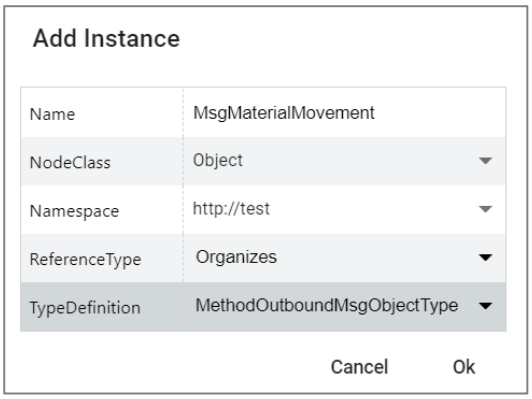
- Your message structure now looks like this, where your handshake will be done on the "Handshake" Variable and the message will be written on the "Content" Variable with its header information written on the "Header" variable.



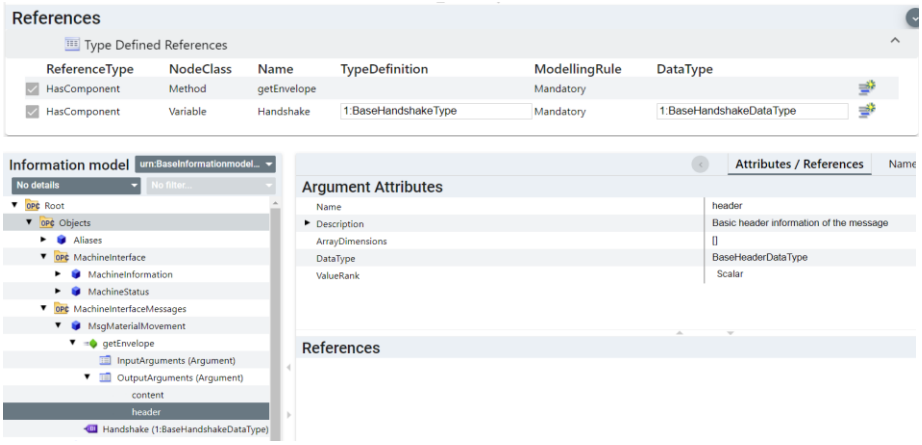
- Don’t forget to specify any abstract data types in your content like TargetQuantity in Order, depending on your use case.

4.4.2 Method-based communication

- For method-based communication choose the ObjectType MethodInboundMsgObjectType or MethodOutboundMsgObjectType

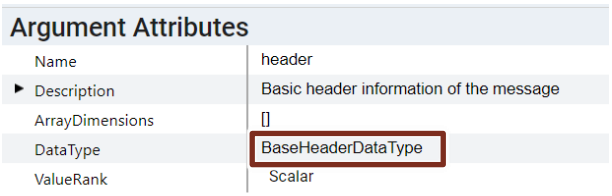


The Outbound message will come with a Handshake variable to initiate the method call if the event happens on PLC. The Inbound message is consisting out of just the method to be called. Both contain a header and a content argument in their methods, where the inbound takes them as input arguments, whereas the outbound returns them as a return argument. The handshake and the header DataTypes are still abstract and need to be specified.

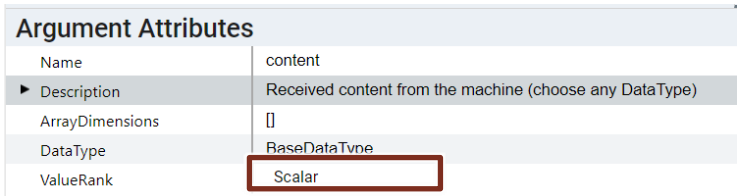


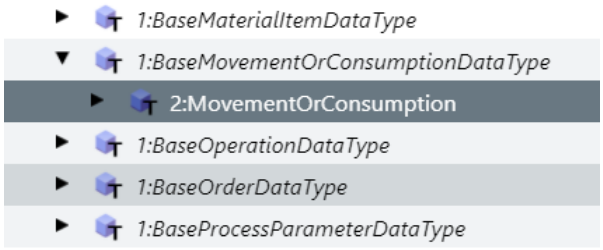
- Specify the Handshake data type as in chapter 4.4.1

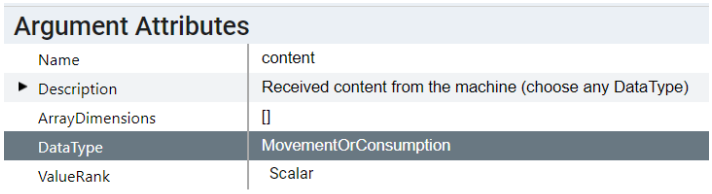
  To specify the Header DataType, go to argument "header" and click on the "BaseHeaderDataType". Choose the required data type (see chapter 4.4.1)



- To specify the Content DataType, go to argument "content" and click on the "BaseDataType". Choose the required content data type (see chapter 4.4.1)



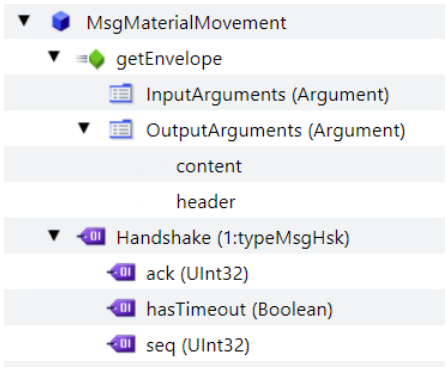




NOTE

If you want to set the types for header and handshake structures generally and not only for each single instance you can specify it in the InterfaceType of the required message object structure before creating the objects. All the instances will then inherit this choice and you don't need to specify it anymore.

- Your message structure now looks like this, where your handshake will be done on the "Handshake" Variable and the message will be transferred by calling the getEnvelope method



- Don't forget to specify any abstract data types in your used DataType. If necessary, copy your DataType (see section 4.6.5) and change it there.

4.5 Create, delete and change the Opcenter interface

To use the out-of-the-box features for shopfloor integration on Opcenter Execution Process, import the namespace urn:MachineIntegrationProcess.Siemens.com.

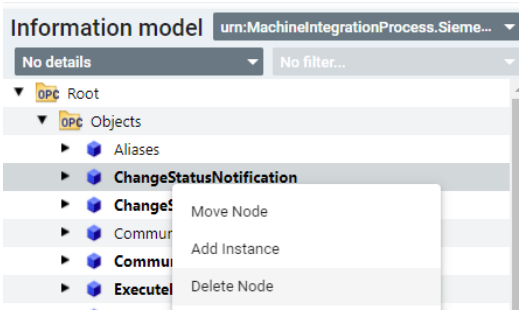
To use the out-of-the-box features for shopfloor integration on Opcenter Execution Discrete, import the namespace urn:MachineIntegrationDiscrete.Siemens.com.

There you can find the existing and working mapped server interface for the basic use cases. You can either:

- Delete message interface that are not sent or received on your machine
- Add some array elements within the already existing structures
- Create a new instance of the provided ObjectTypes to send or receive messages of this type in another parallel channel.

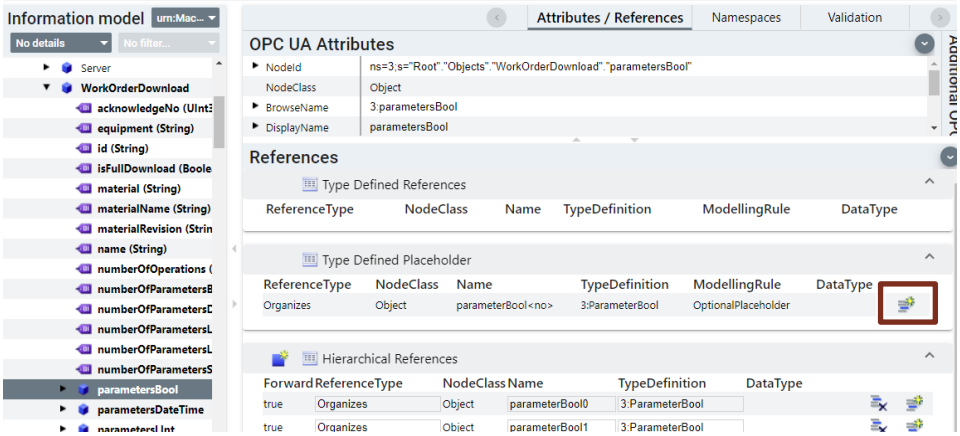
To delete a message interface rightclick the message object and click "Delete Node".

Figure 4-1: Delete Node



To add a new element to an existing array, go to the right parent node that contains the elements and add another Object within the optional placeholder by clicking the symbol in the "Type Defined Placeholder" section.

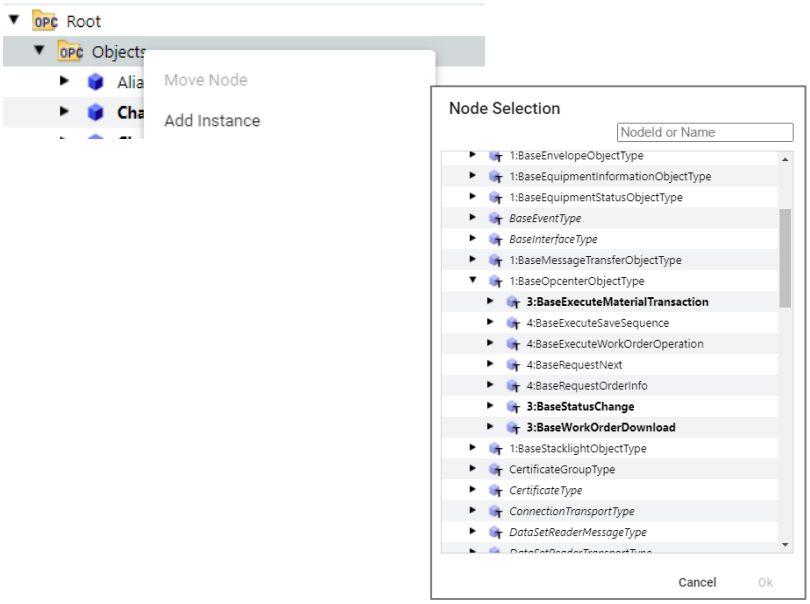
Figure 4-2: Add a new element



To delete an element in the existing array, go to the right parent node that contains the elements and delete the last Object within the optional placeholder by clicking the symbol in the "Hierarchical References" section.

To create a new instance of a message interface, add a new instance, where you desire them and choose out of the provided Opcenter ObjectTypes stored in BaseOpcenterObjectType.

Figure 4-3: Add a new instance



Don’t forget to map new Nodes, create a server interface and load it to the PLC (sections 4.7 and 4.8).

4.6 Create own types

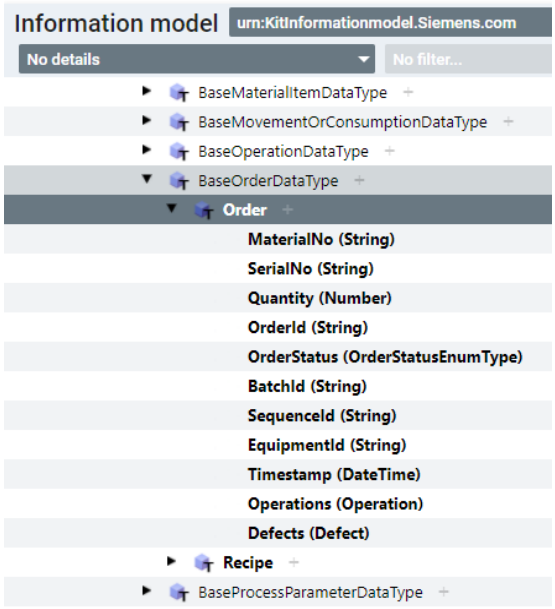
As the provided types are probably not already in the correct form how you need them, you usually need to adjust them. You can either add members or delete members or change Name, DataType or Structure of it. Usually you first change your types while discussing the necessary data to receive and then using it inside the message transfer structure.

In comparison with ObjectTypes, the DataTypes and VariableTypes inside your Objects do not only import the mandatory children into TIA Portal but also the optional. The complete structure will be created as user data type, as TIA Portal is not able to implement optional fields. This means that you need to adjust the Data- and VariableType if you don’t want to use all members as they are. There are several ways how to do that, depending on what you want to do.

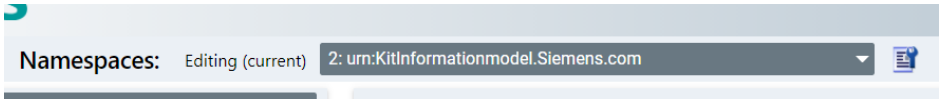
If you want to change Types from the information model kit don’t forget to change the editing namespace, so you are adding things in the right namespace.

4.6.1 Change DataTypes

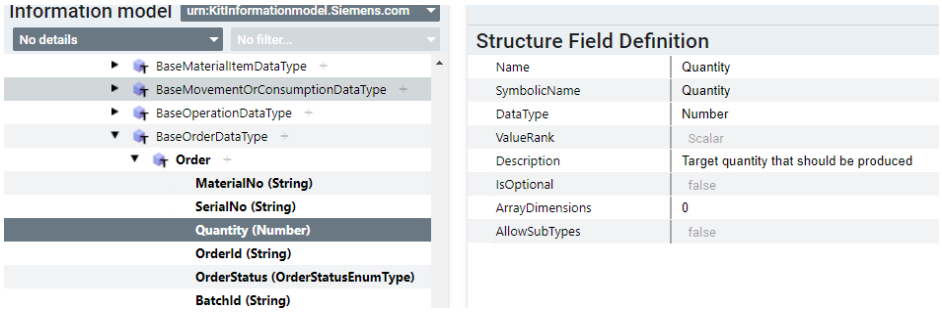
- Go to the DataType that you want to change (i.e. by rightclicking it and chose "Jump To")



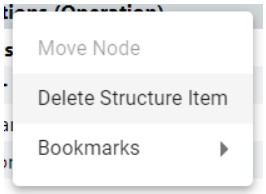
- To change a Datatype, you need to switch to its namespace.



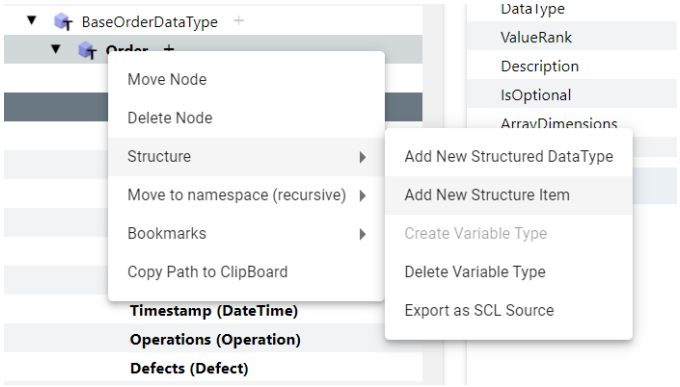
- Change Name, DataType, Description and whether it is defined as Optional by selecting the right Variable and changing it within the Structure Field Definition clicking on the to be changed attribute



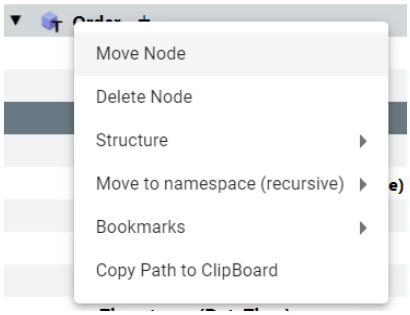
- You can delete an Item by rightclicking it and choosing "Delete Structure Item"



- You can add an Item by clicking the plus or rightclicking the parent structure and choosing "Structur > Add New Structure Item".



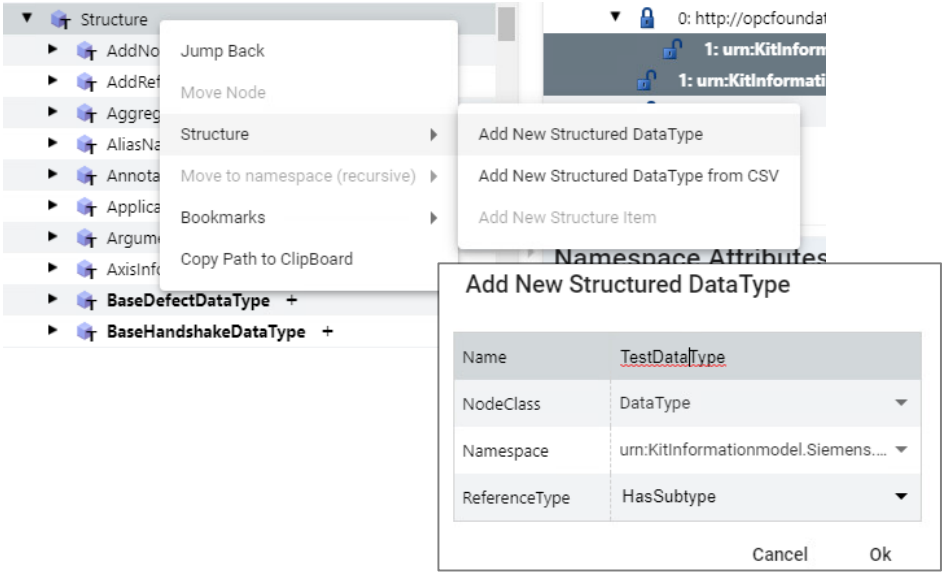
- You can move a DataType from one Parent to the other by rightclicking it and choosing "Move Node".



4.6.2 Create DataTypes

You can just create your own DataType. For this just create a new structures DataType under Root > Types > DataTypes > BaseDataType > Structure
and add the required children by clicking on the plus next to its Node. As soon as you add items, you are setting the abstract property to false.

Figure 4-4: Create DataType



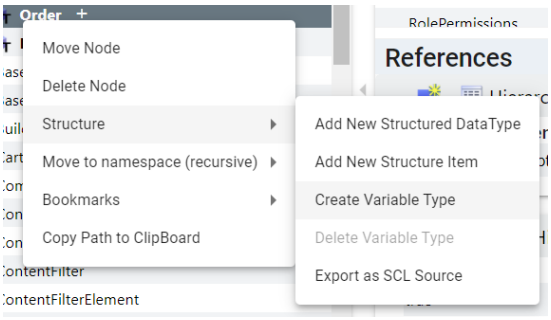
Then you can change the added DataType Items according to section 4.6.1.

4.6.3 Change or create VariableTypes

Generally, the VariableTypes can of course be created from scratch just like the DataTypes. Just avoid changing the name of a Variable itself, as it should be aligned and changed together with the NodeId (if String NodeIds), the BrowseName and the DisplayName.

The easier way to create or change the VariableType is by generating it from the (already existing) DataType. For this, rightclick the required DataType and click "Structure > Create Variable Type". If it is greyed out, it probably already has one.

Figure 4-5: Create VariableTypes



If you want to change the VariableType, it is often easier to change the DataType, delete the VariableType and recreate a new Variable Type again.

NOTE

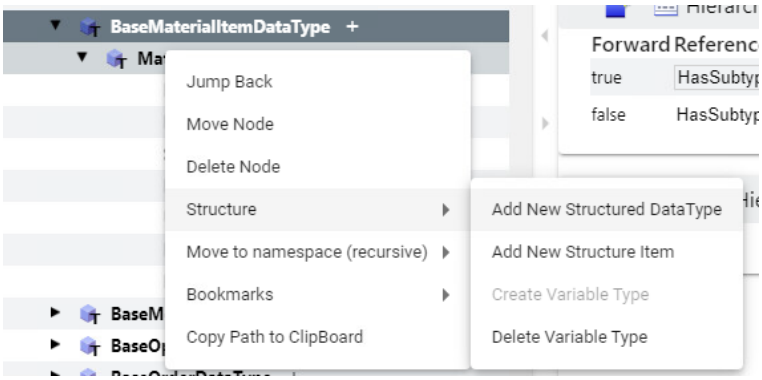
Some Attributes like Description are not automatically taken over and need to be added again. This is also the reason why in this model the VariableTypes (in comparison with the DataTypes) never have Descriptions.

4.6.4 Create DataTypes inheriting other DataTypes

If you find an existing DataType, that you just want to enrich by additional items, you can just create a subtype of it. This subtype will inherit all of the children of the base DataType and add some specific. This is especially helpful for defining a generic part for all the machines and a specific part for only a certain machine or machine class. It will not create a substructure that contains the specific part, but just contains all the inherited or added children flatly. In this way you can save time to create new DataTypes with partly the same content.

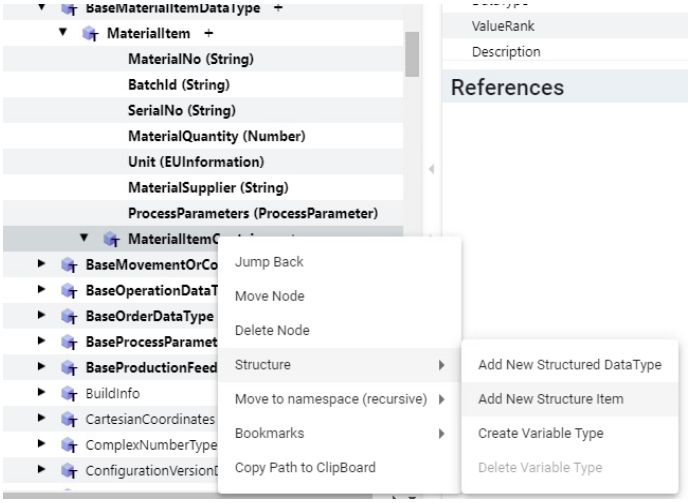
To create a subtype, rightlick the parent DataType and choose "add New Structured DataType". It will automatically contain the Items of the parent DataType.

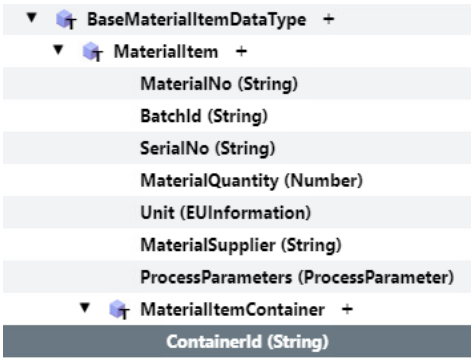
Figure 4-6: Create DataType



Then you can add additional Items to your child DataType. Your subtype will then contain all the parent items plus the own ones.

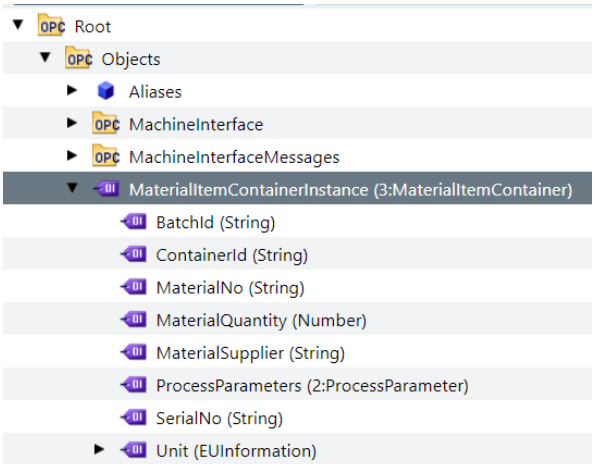
Figure 4-7: Add child DataType





If you now create a VariableType of your new DataType and then instantiate your VariableType it will have automatically instantiate all the inherited members and will look like this:

Figure 4-8: Instance of the new DataType

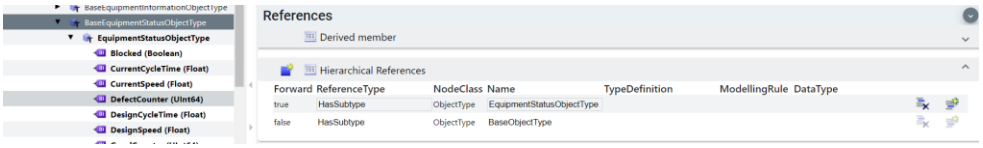


4.6.5 Copy and rename ObjectTypes or Objects

In comparison to Data- and VariableTypes it is comparably easy to copy and rename ObjectTypes, as there is a feature in the here used version of SiOME that allows easy copying ObjectTypes and Variables inside.

To copy an ObjectType, navigate into the parent ObjectType and see the hierarchical References.

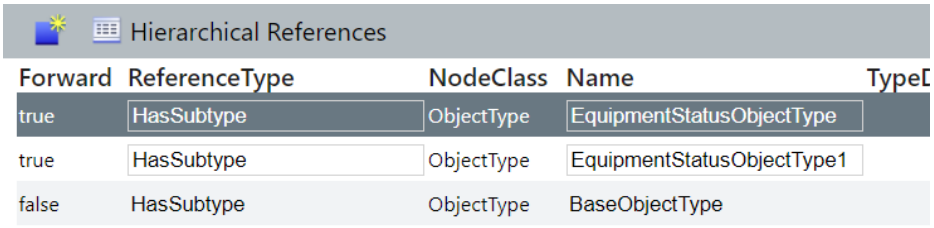
Figure 4-9: Parent ObjectType



Before copying, think of switching to the namespace, where you want to add the new ObjectType to.

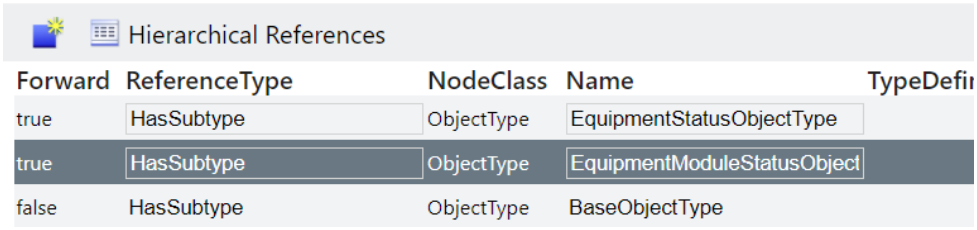
Each subtype is listed and can be copied by clicking this symbol  .

Figure 4-10: Copy



After copying you can directly rename it in the "Name" column.

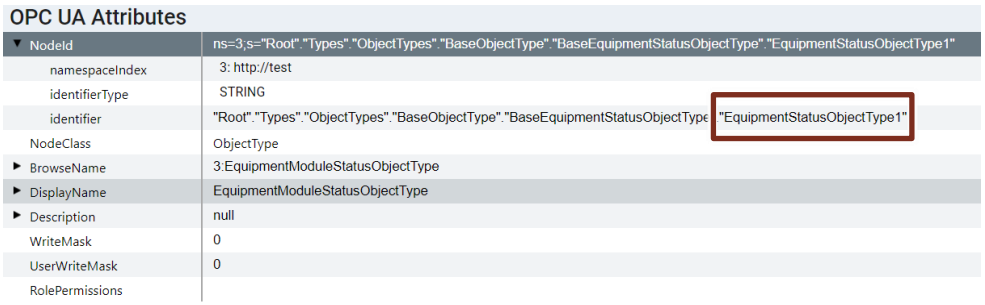
Figure 4-11: Rename



NOTE

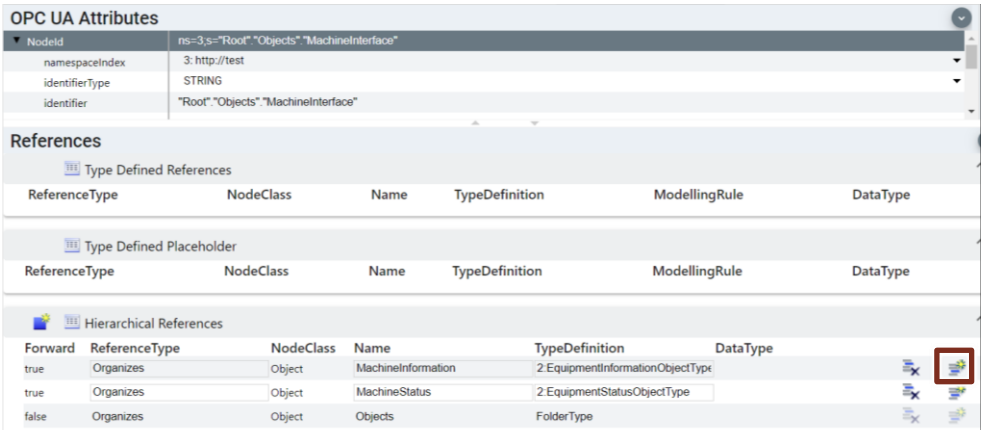
While this changes the DisplayName and the Browsename, it does not automatically set the String NodeId. You need to do that manually if you need it.

Figure 4-12: Rename



Same applies for Objects (instances of ObjectTypes in Root > Objects):

Figure 4-13: Rename



4.6.6 Copy and rename DataTypes or VariableTypes

Sometimes you need to adjust a DataType (i.e. Order from OrderDownload), as you don't need all the members (i.e. for a display of the current processed order) and want to keep the Order DataType for OrderDownload while also creating a similar Order DataType originating from Order, but not inheriting from it for displaying only some of the information.

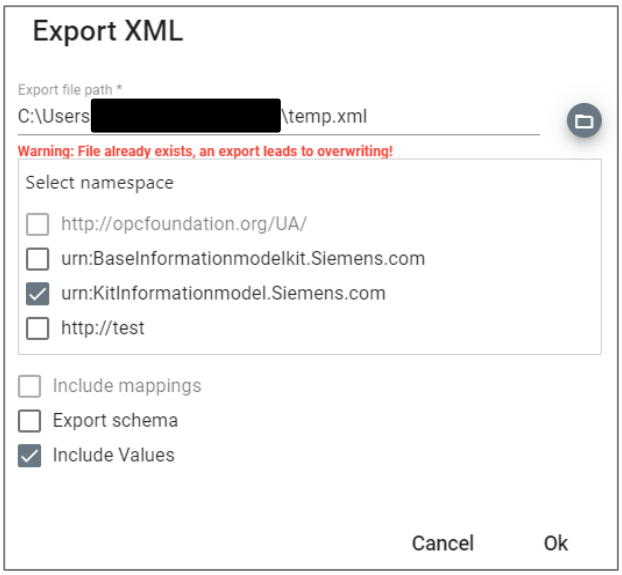
To create your own DataType, basing on an existing one, without creating it from scratch, but still keeping the old one, you need to copy an existing DataType.

It is More difficult than copying ObjectTypes and Objects, as there is currently no out-of-the-box solution to copy or rename them in SiOME directly. Still, there is a work-around how you can still duplicate them.

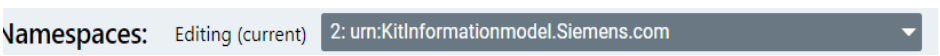
You start at the DataTypes, as the VariableType can later be easily generated out of the new DataType (see section 4.6.3).

To copy a DataType:

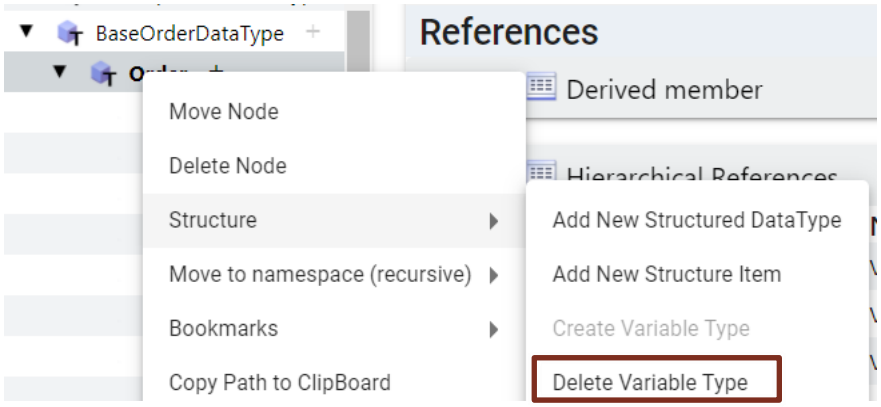
- Be sure that your DataType to keep is already saved or exported. This is already the case if you just want to reproduce the original DataTypes from the provided Information Model Kit. Otherwise export the namespace containing your DataType to keep.



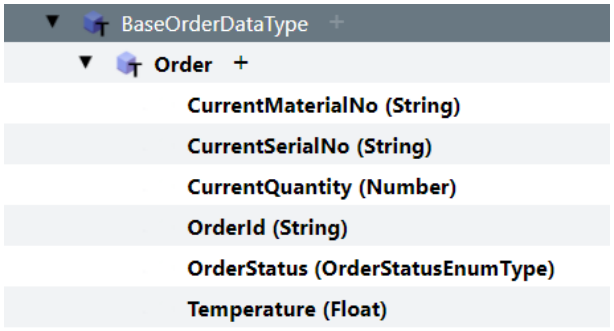
- Switch to the namespace in which the original DataType is located



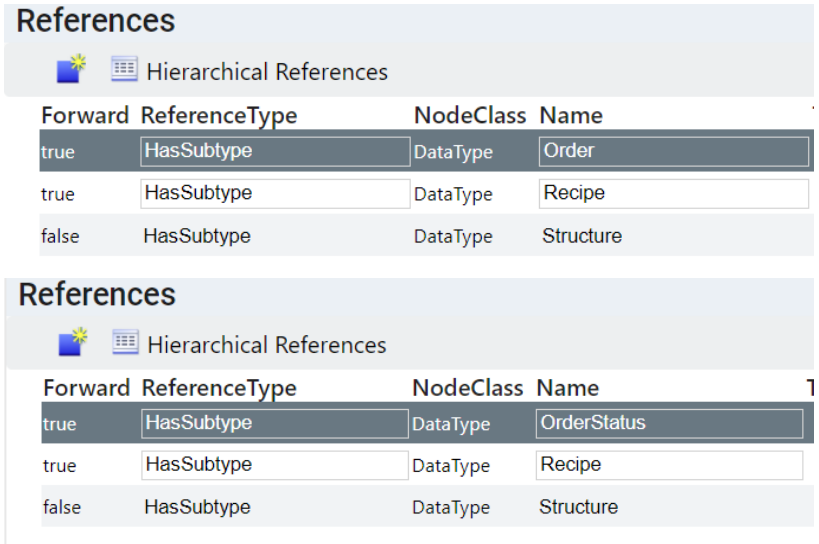
- Delete the VariableType of your DataType



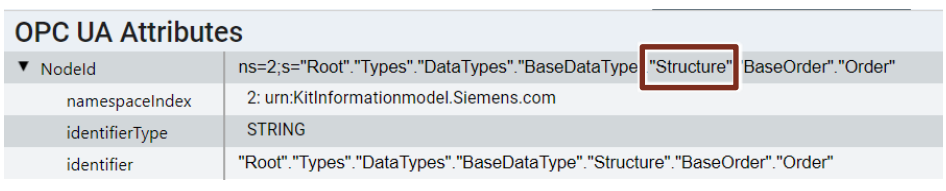
- Change the existing DataType to your preference (delete, add, change members)



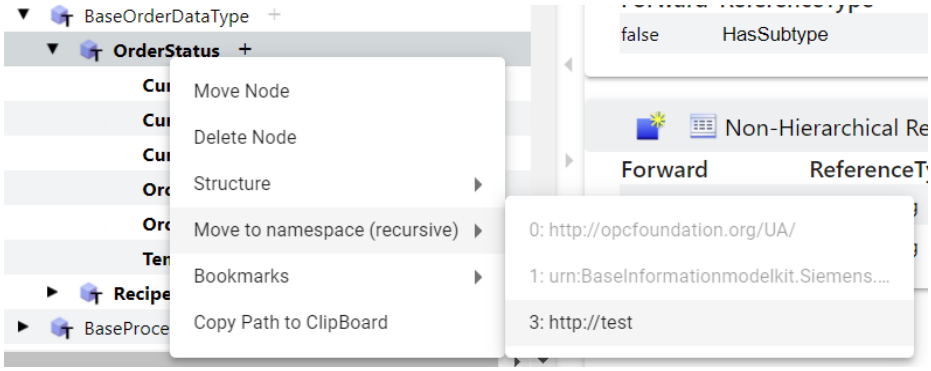
- Rename the DataType



- If using String NodeIds, align it to the new name



- Move the DataType to your or a new namespace

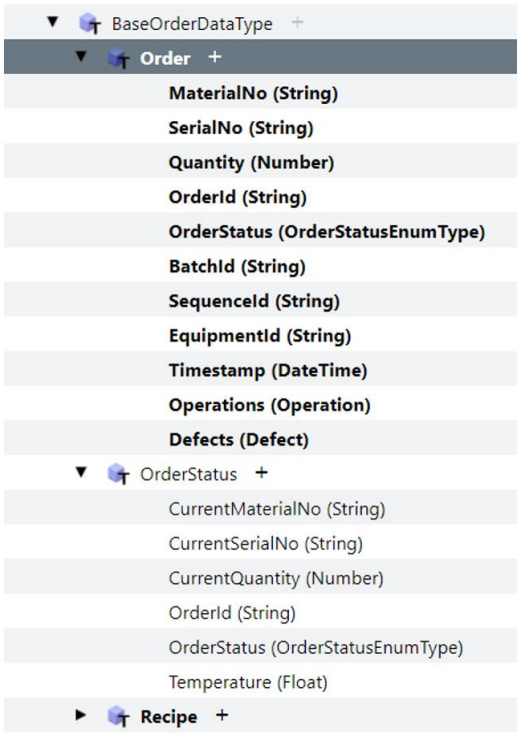


- Import the original namespace or the just saved one with the original DataType.

WARNING

Importing replaces its existing DataTypes. If you changed existing Types within that namespace, the changes are lost. Before importing, be sure that adjusted/changed Types are moved to an own namespace before to avoid data loss.

- This will result in having both, the original and the adjusted DataType.



- Create a VariableType if needed

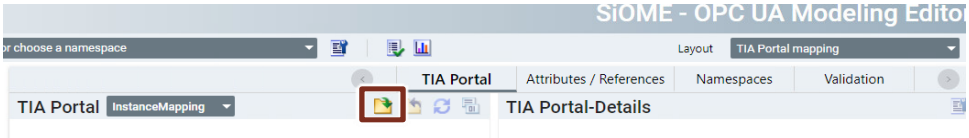
NOTE

Sometimes, it appears as if members are duplicated now after importing the same namespace again. It seems to be a graphical issue. Try to create and delete an additional item and check, if that solved the problem.

4.7 Map to TIA Portal project

To map your information model to your PLC data within the OPC UA server interface, open the corresponding TIA Portal Project in the TIA view.

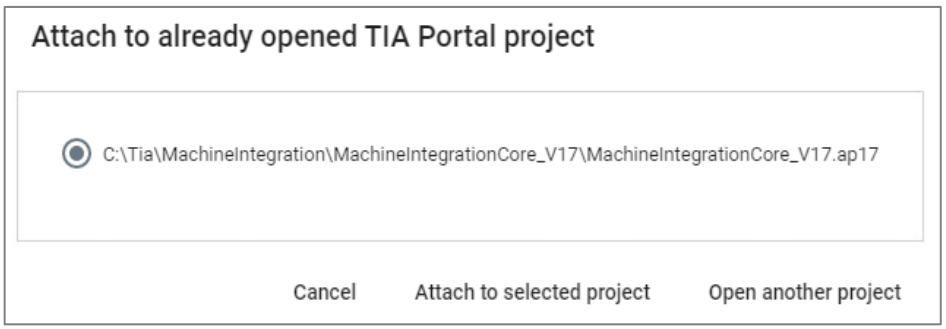
Figure 4-14: Open the corresponding TIA Portal Project



When the project is currently open, you can choose it in the prompted menu and attach to it. Be sure, that your TIA Portal is supporting openness.

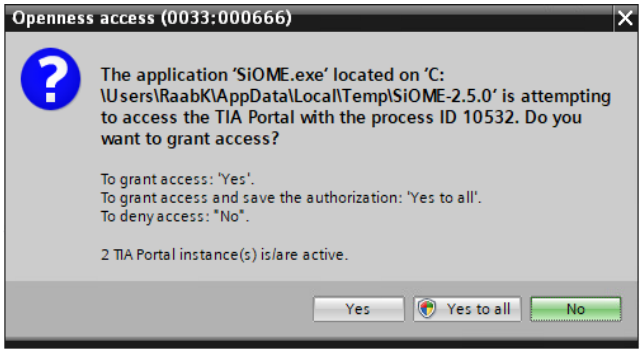
To avoid misbehavior, always compile and save your project before opening or synchronization.

Figure 4-15: Attach to already opened TIA Portal Project



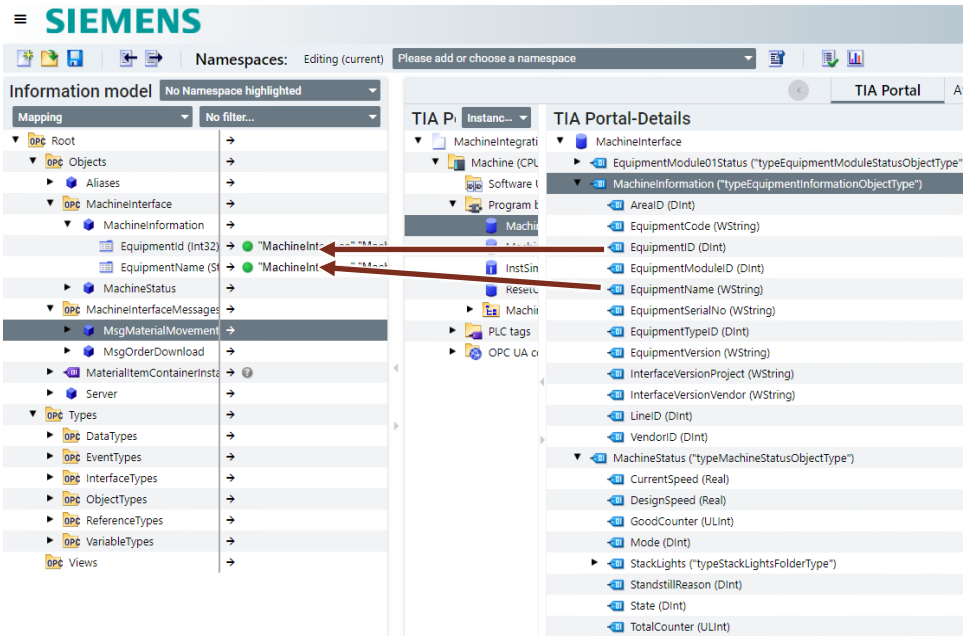
It might be necessary to allow the openness access to your project within TIA Portal before you can see your project in SiOME.

Figure 4-16: Allow openness access to your project



When opened you can see your project tree. Open the DataBlocks that you want to map to your information model. Drag&Drop the available tags from your project to the information model instances to map an existing tag to a variable.

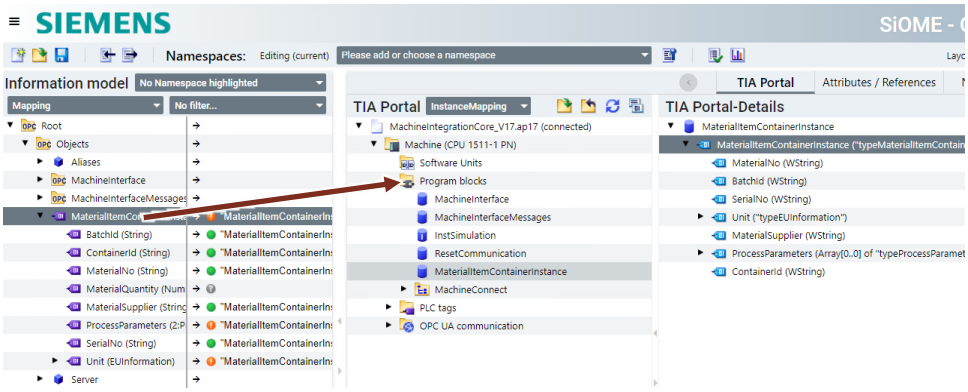
Figure 4-17: Mapping



When your Objects and tags are not yet existing in the TIA Portal project, you can also use the other direction to import it into TIA Portal with mapping included.

For further information about types to be able to use in TIA Project to map to an OPC UA DataType (see the TIA Portal help for informations about OPC UA Data Types and corresponding data types).

Figure 4-18: Import into TIA Portal



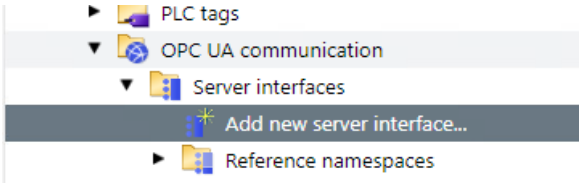
Mapping warnings:

- If there is an abstract DataType left in the structure items, it won't be imported to TIA Portal, as no Size and structure info is available. Eventually the Parent type from SiOME differs from the UDT in TIA Portal, which results in a warning.
- Timestamps in TIA Portal should be in LDT if mapped to DateTime, otherwise it results in a warning.
- Arrays are not imported as an array of the correct length but of length one. Eventually the Parent type from SiOME differs from the UDT in TIA Portal, which results in a warning.
- Enumeration in SiOME can be mapped to integers in TIA Portal, which works, but results in a warning.

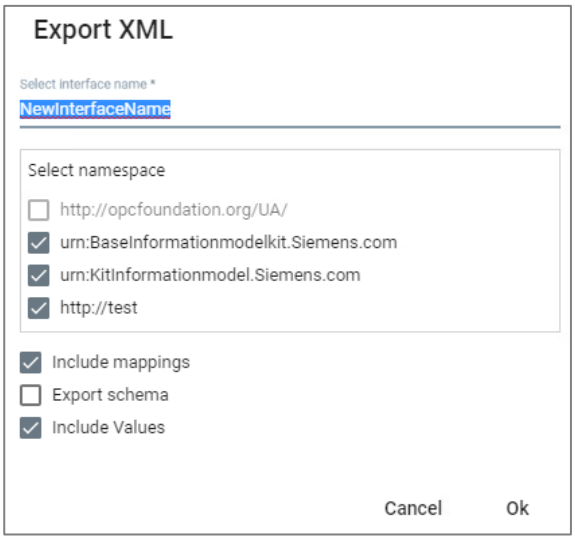
4.8 Create OPC UA Server interface in TIA Portal

If you are done with mapping and importing, you can load an OPC UA server interface from your SiOME NodeSet on your PLC. Otherwise it won't influence to the PLC nor to its server interface.

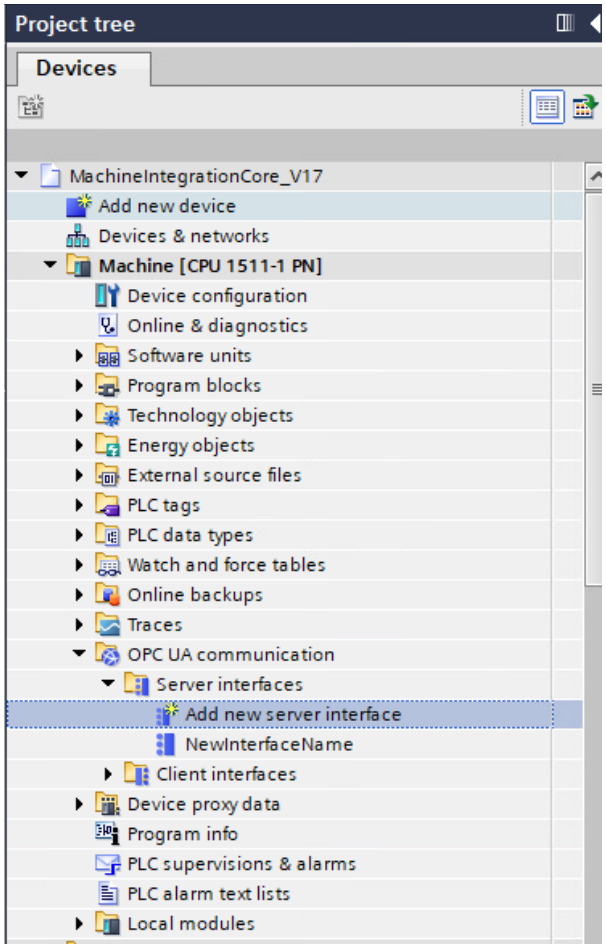
- For this, open "OPC UA communication > Server interfaces" in SiOME and create a new server interface

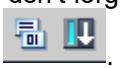


- Only export the namespaces that you need and the ones that you have dependencies to, as the number of nodes to store on the OPC UA Server is restricted depending on your type of PLC. Keep in mind that you can move nodes to other namespaces if you want to decouple parts from each other.



- After creating it in SiOME, you can find it in TIA Portal under "OPC UA communication > Server interfaces"



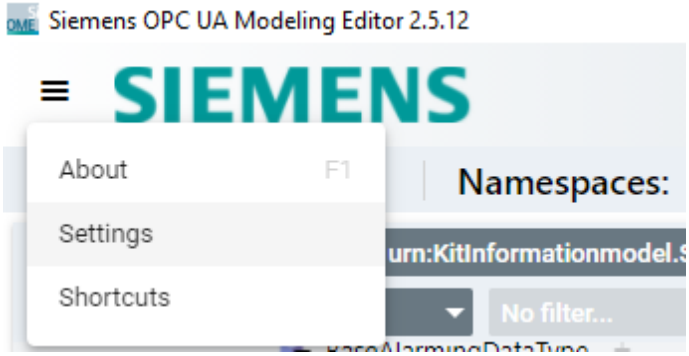
- After that, don’t forget to compile and load your program with your new Server interface to your PLC  .

5 Frequently asked questions

How can I create NodeIds in string-format / numerical format?

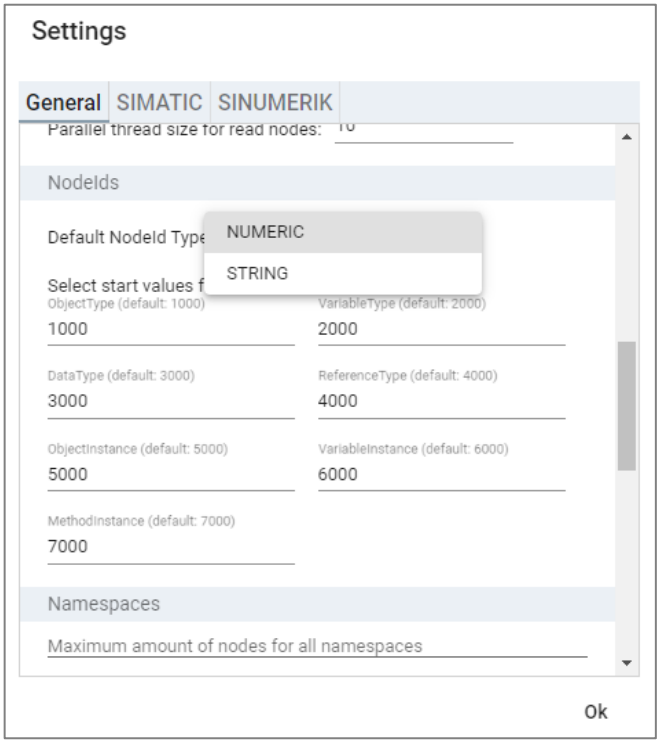
You can change the format with which Nodes are automatically created in the settings. For this, go to settings:

Figure 5-1: Open Settings



Under "General" in the section "NodeIds" you can switch the default NodeId type to the required setting.

Figure 5-2: Settings

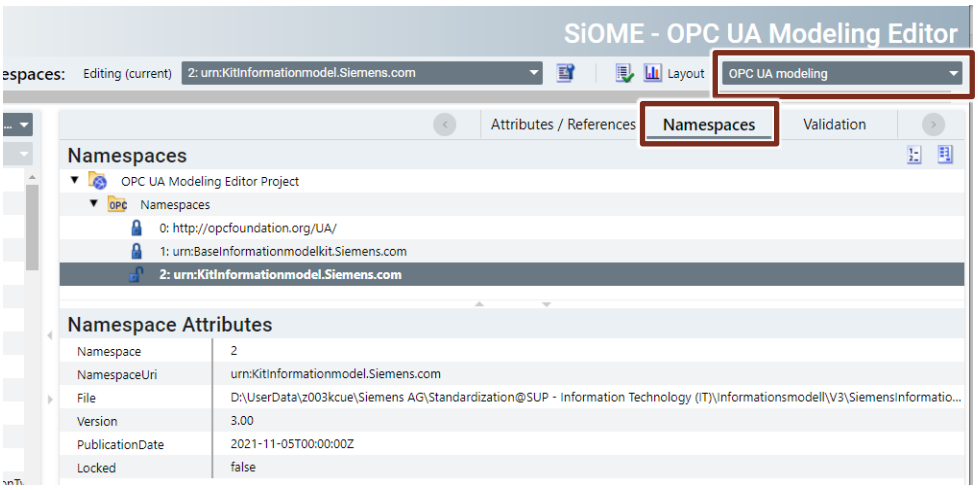


After the creation of a NodeId you can also change the id, but you need to define it by yourself. It is usually easier to delete the node and create it with the required setting again.

How can I change the details of a namespace

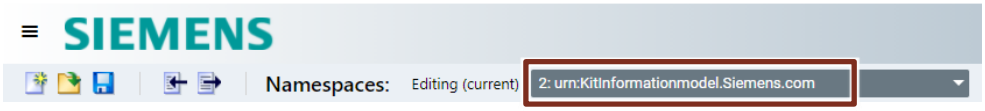
Unlock the namespace that you want to change. Go to the View OPC UA modelling and navigate to Namespaces.

Figure 5-3: Open Namespaces



Select the required namespace. You can change the NamespaceUri, the Version and the PublicationDate by clicking inside the according field and changing the value. You can unlock it by selecting the namespace in the Editing (current) dropdown menu.

Figure 5-4: Select Namespaces



You can change the namespace number by changing the order in the "change order view:

Figure 5-5: Change namespace number

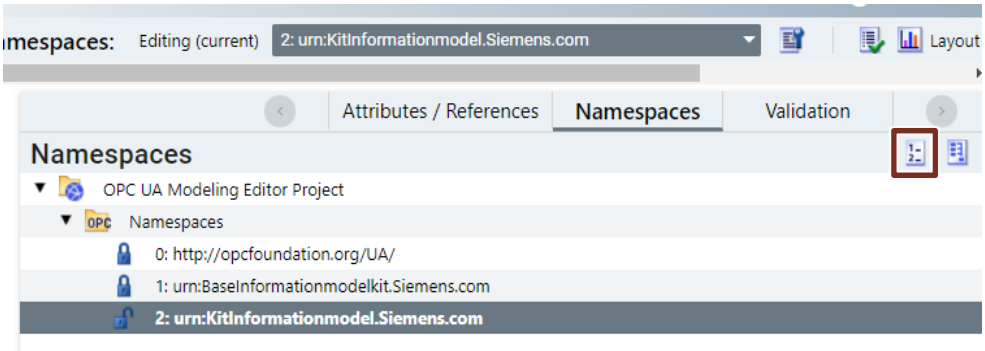
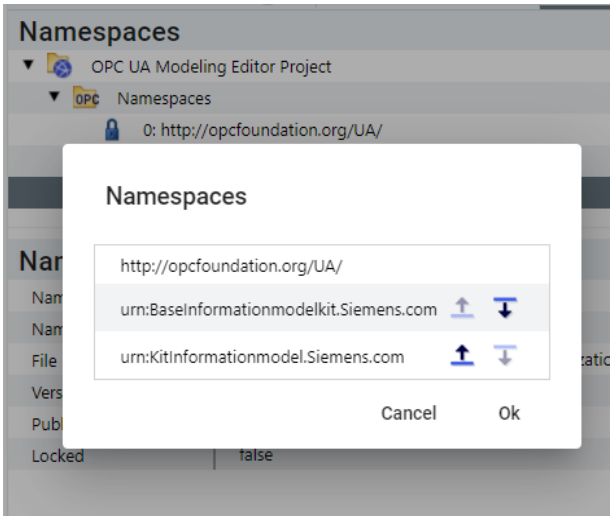


Figure 5-6: Change namespace number



How can I model arrays?

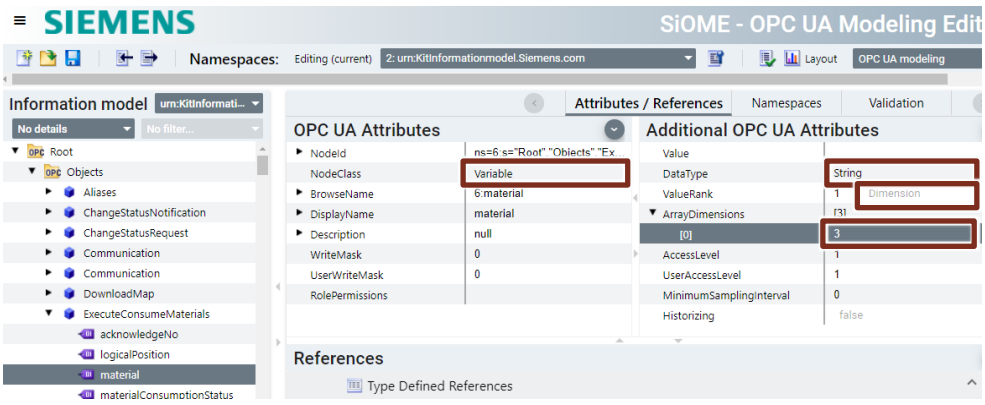
There are two ways to model arrays which differ in the way to access the data:

- As a Variable with Dimensions of a DataType
  The array elements within one dimension of a variable are no own variables and therefore cannot be seen as own nodes within the object tree. As the size and structure of the DataType is fixed, also the the size and structure of the Array is fixed. A Variable of a DataTypes with a dimension can only be accessed if reading/writing of complex data types is supported.
- As an Object or Variable containing Objects or Variables
  Objects and Variables can have several children nodes (Variables or Objects) of the same structure and type. Each of the child nodes are displayed in the object tree and can be read/written individually. If there is a corresponding array implemented on machine side, each of the Variables needs to be mapped individually.

Adding an array as dimensional variable

To add an array as dimensional variable, create a variable of BaseDataVariableType VariableType and the DataType of which the array should be of. Change the ValueRank to "Dimension" and choose "1 Dimension" for a one-dimensional array. Type in the length of the array.

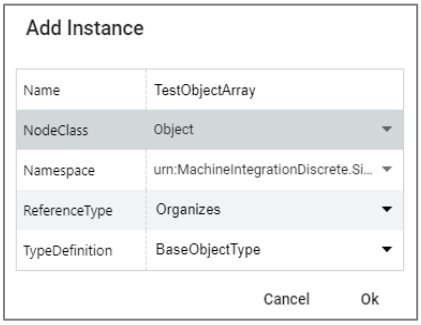
Figure 5-7: Adding an array as dimensional variable



Adding an array as Object containing Objects or Variables

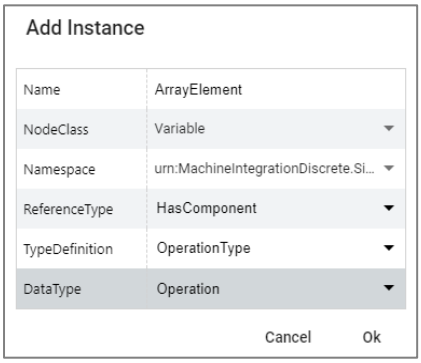
If you want to add an Object of the same Variable structure, create a parent Object of BaseObjectType.

Figure 5-8: Create a parent Object of BaseObjectType



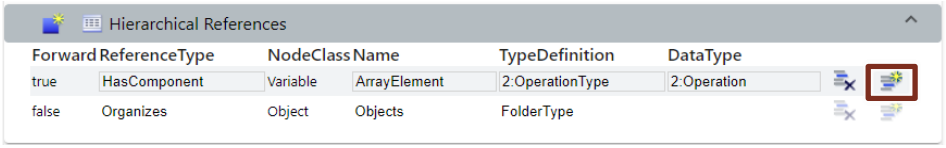
Within that Object create an Object with Variables or a Variable of your required DataType and VariableType, e.g. with Operations.

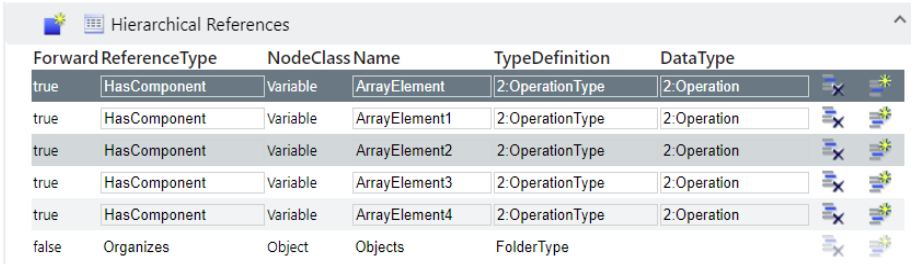
Figure 5-9: create an Object with Variables or a Variable



Go to the section Hierarchical References and copy the ArrayElement as often as you need it

Figure 5-10: Copy ArrayElement



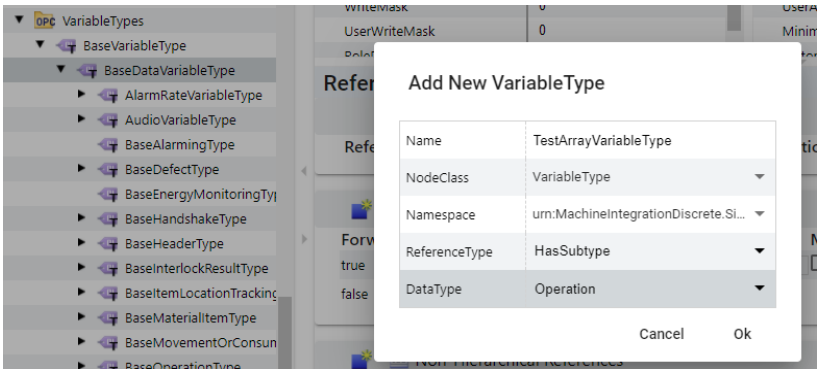


You can rename the first element in column "Name" or delete it with a click on the delete symbol next to the create symbol.

Adding an array as Variable inside the VariableType containing Objects or Variables

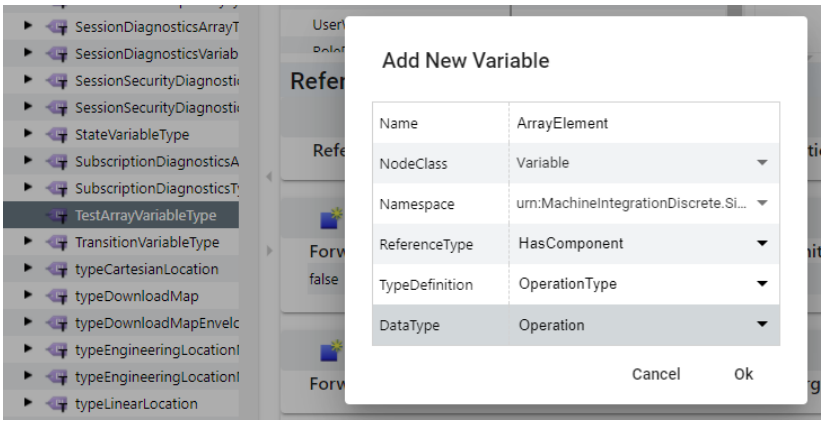
Create a new VariableType as a parent Node for the Array Elements. Use the DataType of the ArrayElements.

Figure 5-11: Create new VariableType



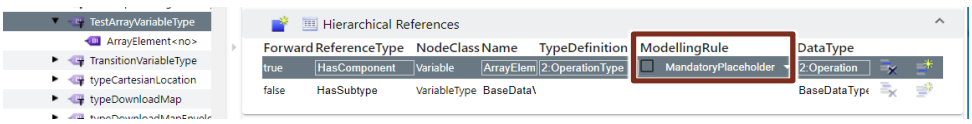
Add a Variable to the VariableType of the VariableType and the DataType of the ArrayElements.

Figure 5-12: Add new Variable



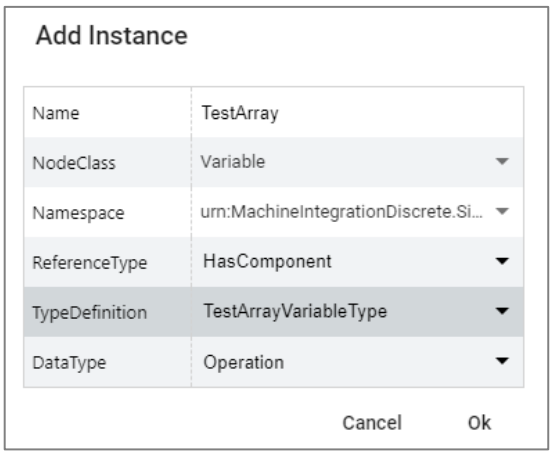
Select the parent node and set the "ModelingRule" of the added Variable to "MandatoryPlaceholder".

Figure 5-13: Set the "ModelingRule"



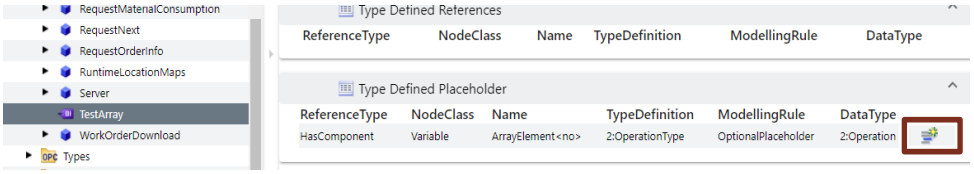
Whenever you create an Instance of the array VariableType, you can add an arbitrary number of ArrayElements.

Figure 5-14: Add Instance



For this, add a Variable instance of the created VariableType and click on the "instantiate" button in the area "TypeDefinedPlaceholder".

Figure 5-15: Instantiate



Why are members missing in my TIA Portal Project after I imported Nodes?

Maybe your members are optional. TIA Portal does not support optional members in DataTypes. DataTypes have a defined length and structure. If a member within a DataType of a Variable is optional, this member will not be included in the import. If the DataType that you want to import into your TIA Portal Project can vary, it is recommended to create a new DataType as in section 4.6 described.

Maybe your members are abstract. TIA Portal does also not support abstract DataTypes. Abstract DataTypes do not contain any own members. They can be used to group subtypes or even to inherit members. A Variable or Object of an abstract DataType will not be imported to the TIA Portal Project.

What's the difference between VariableType and DataType?

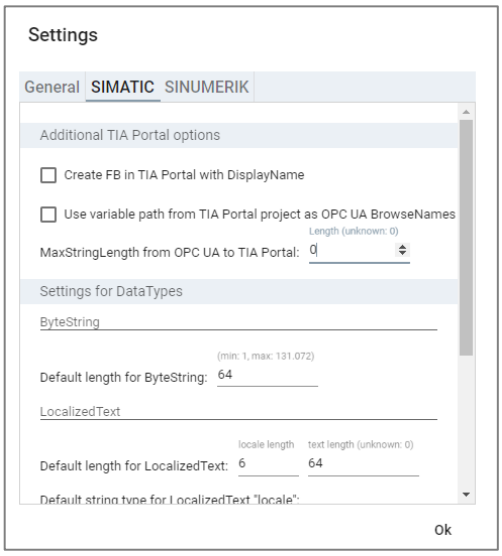
For further information about the difference also see
https://support.industry.siemens.com/cs/ww/en/view/109773628

The DataType of a Variable defines the size and structure of the Value inside the Variable. If you write a variable, you write the complete value. If this is a complex DataType, the complex structure will be written, not only a single member of the structure. To access a single member of the structure, you need to browse the Variable through the VariableType and read or write the basic Variable within the complex VariableType. A Variable without a specific VariableType are still accessible. A variable with a complex VariableType is furthermore browsable and the members are accessible individually.

What's the length of a String?

The length of a String in the information model is undefined. The standard length of the WString that is created when importing a String in TIA Portal can be defined in the settings under "SIMATIC" in the section "Additional TIA Portal options", where you can change the MaxStringLength from OPC UA to TIA Portal.

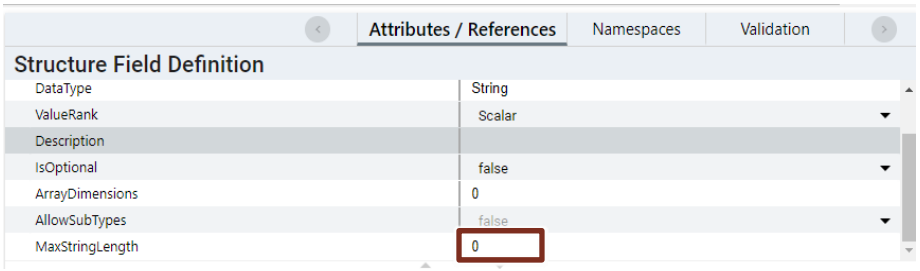
Figure 5-16: Settings MaxStringLength



If the length is unknown (0), it automatically restricts the String to a length of 256 characters. You can still change the length in the TIA Portal project. If a String is accessed with too many characters, a bad response will be returned. The maximum WString length of a SIMATIC S7-1500 PLC is at around 16200.

You can also set the maximum length of a specific String item of the interface within a structure DataType.

Figure 5-17: MaxStringLength



6 Additional Information

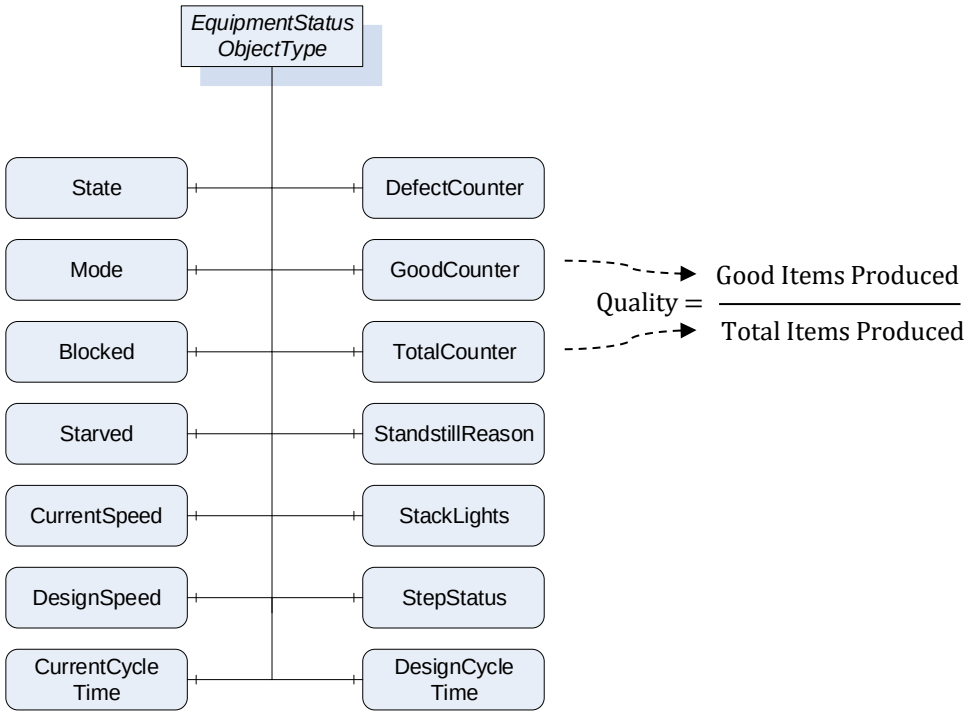
6.1 KPI calculation using EquipmentStatusObjectType

6.1.1 Basic KPI "Quality"

Some KPIs such as the quality can easily be calculated using directly the continuous machine data from the EquipmentStatusObjectType.

The quality is defined as "Good Items Produced" divided by "Total Items Produced".

Figure 6-1: EquipmentStatusObjectType



6.1.2 Complex KPI "Overall Equipment Effectiveness" (OEE)

The OEE calculation is shown in the following formula.

Figure 6-2: OEE calculation

$$OEE = \text{Availability} * \text{Performance} * \text{Quality}$$

$$= \frac{\text{Actual Production Time}}{\text{Planned Production Time}} * \frac{\text{Actual Production Rate}}{\text{Ideal Production Rate}} * \frac{\text{Good Items Produced}}{\text{Total Items Produced}}$$

$$= f(\text{Mode}, \text{State}) * \frac{\text{CurrentSpeed}}{\text{DesignSpeed}} * \frac{\text{GoodCounter}}{\text{TotalCounter}}$$

optional

$$\frac{\text{DesignCycle Time}}{\text{CurrentCycle Time}}$$

Quality and Performance can be directly read and calculated from the variables of the EquipmentStatusObjectType.

The quality is defined as "Good Items Produced" divided by "Total Items Produced". The performance is defined "Current Speed" divided by "Design Speed".

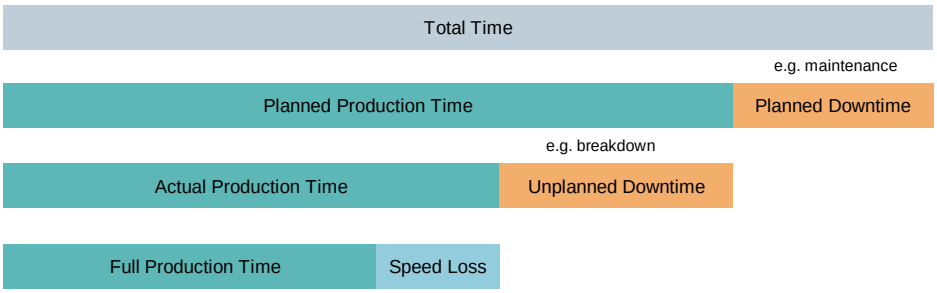
The calculation of the availability requires the definition and the acquisition of time related information, in seconds or minutes.

For example, the Planned Production Time is an 8 hour shift, thus 480 minutes.

Due to planned downtime (30 minutes) and unplanned downtime (60 minutes) the actual production times was reduced to 390 minutes. The availability is defined as "Actual Production Times" divided by "Planned Production Time". In our example 390 minutes / 480 minutes = 0,8125 or 81,25% Availability.

Unfortunately the availability cannot directly be calculated from the variables provided by the EquipmentStatusObjectType. However, most OEE- or Performance Managements Systems use databases in the background and have the ability to create record-sets with time stamps for mode- or status-changes of a machine. These record-sets can then be used to calculate the unscheduled downtime and in consequence the actual production time.

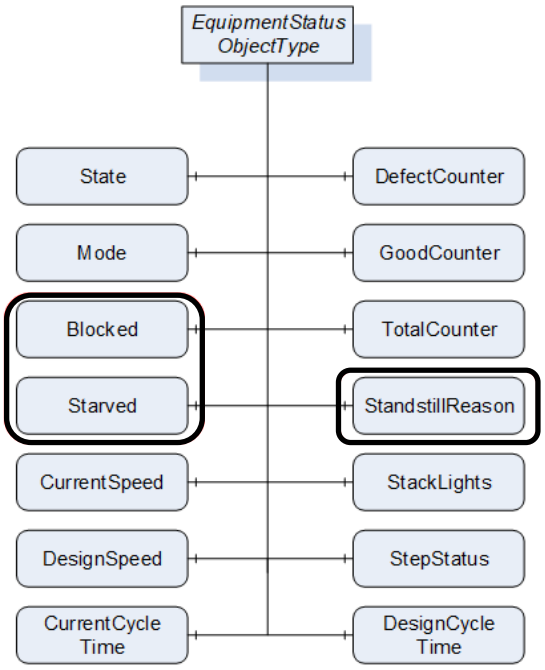
Figure 6-3: Time Model



The definition of a time model, seen in the picture above is a common feature of a performance management application and supports the grouping of machine states and modes to certain time categories. As someone can well imagine, the acquisition of planned downtimes, such as planned maintenance or lunch-breaks, cannot be retrieved from the machine itself. Instead these data must be provided from a shift-calendar or a maintenance planning tool and are therefor not part of the EquipmentStatusObjectType.

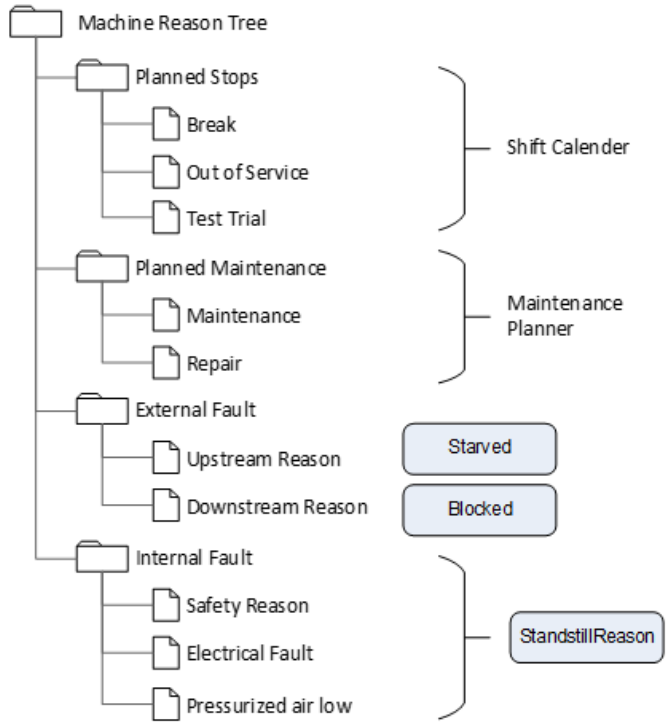
The are 3 more variables (Blocked, Starved, StandStillReason) provided by the interface, that allow a more precise analysis of unplanned downtimes. They are not necessary for OEE calculation, but can be used for statistic evaluations of stoppage reasons and can support root cause analysis of a machine.

Figure 6-4: EquipmentStatusObjectType



The following picture shows a machine reason tree, which is a typical engineering element of a downtime management application. The reason tree groups all machine downtime reasons into categories. The leaves of the categories represent the detailed stoppage reason. By mapping the appropriate variables from EquipmentStatusObjectType_to the leaves of an machine reason tree, the performance management system can create time base record sets.

Figure 6-5: Machine Reason Tree



The categories of the machine reason tree are linked to a time category in the time model. For example, time category "Unplanned Downtime" is dissolved by External Fault and Internal Fault.

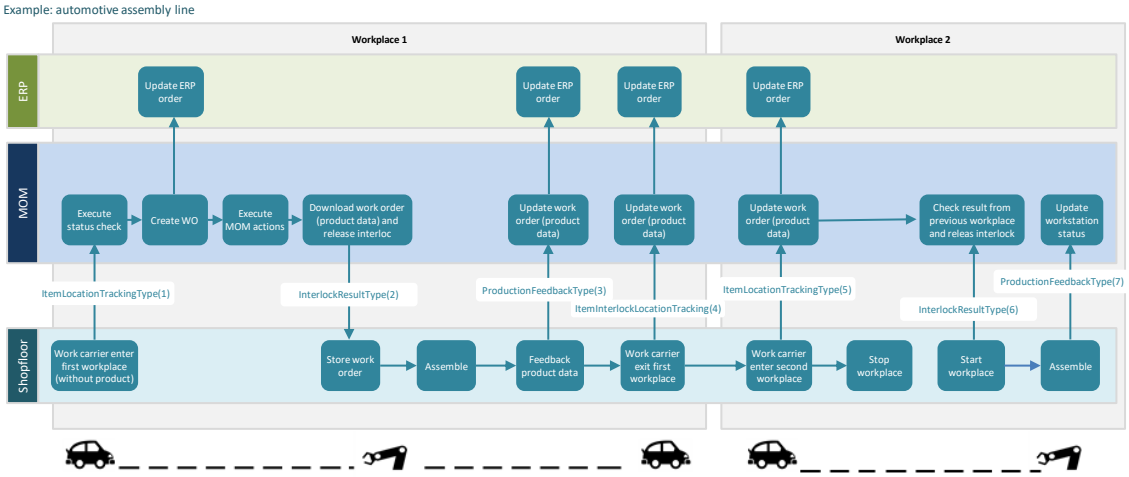
6.2 Item location tracking and interlocking a manufacturing process

In this chapter we want to give an example how the objects

- ItemLocationTrackingType
- InterlockResultType
- ProductionFeedbackType

can be used in an automotive scenario. Keep in mind, that the scenario is simplified, and we will discuss only the communication between shopfloor and MOM.

Figure 6-6: Automotive scenario



The car is entering the workplace 1, which is the first workplace in the line. The car is automatically scanned when entering the workplace an identifying itself to the MOM system by sending the message ItemLocationTrackingType(1).

In our example the body of the car must be hold and a workorder in MOM created before the assembly process can be started. Therefore, we will set the Interlock to True. By the variable LocationTracking, which must be prior defined, we will tell the MOM system, why the interlock was set. The variables XYZposition are not required in this scenario and can remain empty or unused.

Table 6-1

Name	Type	Description
ItemLocationTracking DataType	Structures	
SerialNo	String	Unique Id of the car
LocationId	String	Id which identifies the location "enter workplace 1"
Timestamp	DateTime	Timestamp of the physical event
Interlock	Boolean	True: Execute workorder creation at MOM level before continuing
LocationTracking	LocationTrackingEnumType	Init_3 Item is entering the station and must be initialized. A workorder must be created and downloaded
Xposition	Int32	(Not required – alternative to LocationId)
Yposition	Int32	(Not required – alternative to LocationId)
Zposition	Int32	(Not required – alternative to LocationId)
MaterialId	String	(Not required – alternative to SerialNo)

Name	Type	Description
BatchId	String	(Not required – alternative to SerialNo)

In the next step, MOM will create a workorder and release the interlock at shopfloor with the message InterlockResultType(2).

Table 6-2

Name	Type	Description
InterlockResult DataType	Structure	
SerialNo	String	Unique Id of the car
LocationId	String	Id which identifies the location "enter workplace 1"
Timestamp	DateTime	Timestamp
InterlockRelease	Boolean	True (release car body and continue assembly)
LockReason	String	if interlock release = FALSE, for example when the SerialNo is unknown, MOM cannot create a workorder and release the interlock
Operation	OperationType	Not required
Order	OrderType	Contains the full content of the workorder and all information needed at shopfloor level

The InterlockResultType contains the OrderType with all information required at the shopfloor level for the execution of the assembly order, including operations, materials, process parameters and of course some header information such as OrderID, targetSerialNo, equipmentID… The OrderType will be explained in a separate example in more detail.

Once the workorder is stored at shopfloor and the interlock is released the assembly process can start. When the car is exiting the workplace 1 two messages are send to MOM.
The first message ProductionFeedbackType(3) is meant to update MOM and ERP about the actual manufacturing process status and to update MOM and ERP with material consumptions.

Table 6-3

Name	Type	Description
ProductionFeedback DataType	Structures	
MaterialId	String	Identifier of the material of the car body
SerialNo	String	Unique Id of the car
Quantity	Number	1
OrderId	String	Order number
BatchId	String	Batch Id (not required)
SequenceId	String	SequenceId of the car in the sequence of cars for the current week
EquipmentId	String	Workplace 1
Timestamp	DateTime	Timestamp when parts where produced
Operations	OperationType	Array of operations
QualityStatus	QualityStatusEnumType	Product quality
Defects	DefectType	Array of defects
OrderStatus	OrderStatusEnumType	Curernt status of order

The second message ItemLocationTrackingType(4) is meant to inform MOM that the car is leaving the station so MOM knows where the car is located now. The variable interlock will be False, no interlock required when leaving the station.

The car is now entering the workplace 2, which is the second workplace in the line. The car is automatically scanned when entering the workplace and identifying itself to the MOM system by sending the message ItemLocationTrackingType(5).

Table 6-4

Name	Type	Description
ItemLocationTracking DataType	Structures	
SerialNo	String	Unique Id of the car
LocationId	String	Id which identifies the location "enter workplace 2"
Timestamp	DateTime	Timestamp of the physical event
Interlock	Boolean	True: check result of previous workplace before continue assembly
LocationTracking		
Xposition	Int32	(Not required – alternative to LocationId)
Yposition	Int32	(Not required – alternative to LocationId)
Zposition	Int32	(Not required – alternative to LocationId)
MaterialId	String	(Not required – alternative to SerialNo)
BatchId	String	(Not required – alternative to SerialNo)

Note, that every time a car body is entering a new workplace the MOM system is requested to check the results of the previous operations or quality information. Defects, quality issues can be detected at any time during the production and will be stored against the SerialNo of a car body in MOM. This check will assure that only good items will be processed or at least the information is stored for potential rework or reject.

Once the car SerialNo is successfully checked MOM will send a message InterlockResultType (6) to release the car body and start the assembly at workplace 2.

Table 6-5

Name	Type	Description
InterlockResult DataType	Structure	
SerialNo	String	Unique Id of the car
LocationId	String	Id which identifies the location "enter workplace 2"
Timestamp	DateTime	Timestamp
InterlockRelease	Boolean	True (release car body and continue assembly)
LockReason	String	if interlock release = FALSE, for example when the SerialNo requires rework, e.g. missing material from workplace 1
Operation	OperationType	Can be used to download parameters and materials required at workplace 2
Order	OrderType	Not required

Before the car body exits the workplace 2, a message ProductionFeedbackType(7) is sent in order to update MOM and ERP about the actual manufacturing process status and to update MOM and ERP with material consumptions.

7 Appendix

7.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers
– ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:

siemens.com/SupportRequest

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:

siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:

support.industry.siemens.com/cs/ww/en/sc/2067

7.2 Industry Mall

The Siemens Industry Mall is the platform on which the entire siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:
mall.industry.siemens.com

7.3 Links and literature

Table 7-1

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109814835
\3\	

7.4 Change documentation

Table 7-2

Version	Date	Modifications
V3.0	06/2023	First released version