OPC UA PubSub with SIMATIC S7-1500 based on MQTT

SIMATIC S7-1500 / OPC UA PubSub / MQTT / LOpcUa

https://support.industry.siemens.com/cs/ww/en/view/109797826

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 at: https://www.siemens.com/industrialsecurity.

Table of Contents

1 Introduction

1.1 Overview

With part 14 of the OPC UA specification (\3\), the communication mechanism "PubSub" was introduced. This mechanism is not based on the Client–Server principle as in conventional OPC UA communication (OPC 10000-4), but instead uses the "Publish/subscribe principle".

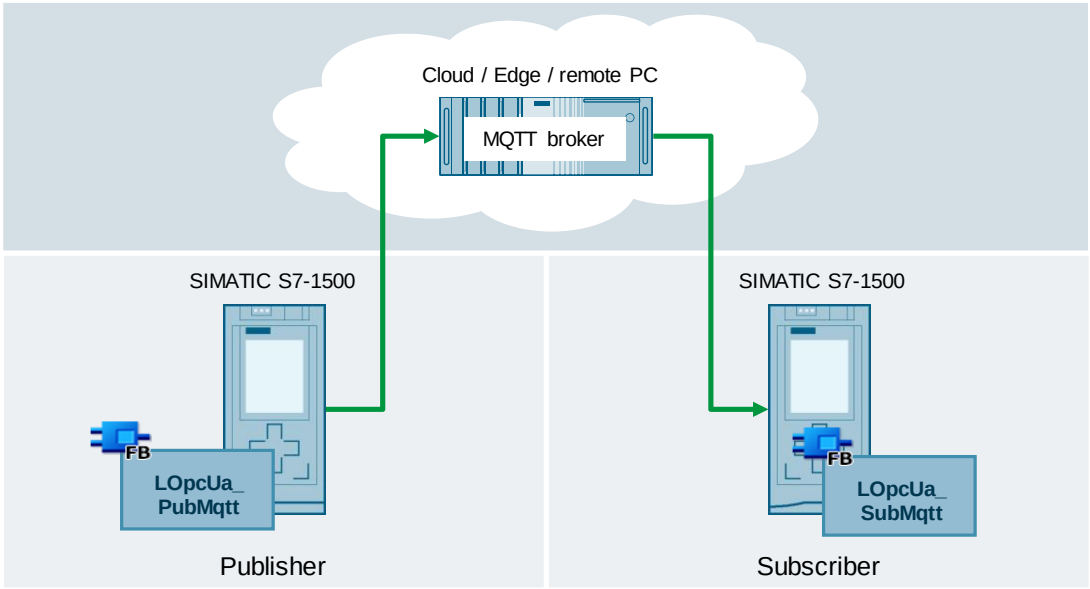
The following scenarios arise when using OPC UA PubSub:

- Configurable peer-to-peer communication between controller and PLC or HMIs: The stations are not directly connected with each other and a configuration in order to make the stations known to each other is not necessary. It can be a point-to-point connection or data distribution to many receivers (multicast).
- Logging on multiple systems: Sensors or actuators can write logs into a monitoring system, an HMI, an archive system for later retrieval.
- OPC UA Servers that represent services or devices can stream data to applications hosted in the cloud. For example, back-end servers, big data analysis for system optimization, and predictive maintenance.

With the TIA library "LOpcUa", Siemens provides function blocks for the implementation of OPC UA PubSub for SIMATIC S7-1500 controllers. Both the Publish and Subscribe blocks of this library implement PubSub via MQTT and use the transport protocol mapping "UADP" for this purpose.

This application example shows you how to use the blocks of the library. Basically, any other communication partner that supports
OPC UA PubSub via MQTT can be used.

Figure 1-1: Schematic structure of the application example



OPC UA PubSub via MQTT offers you the following advantages:

- Requires standard network equipment and only an MQTT Broker as an additional software component.
- All controllers of the SIMATIC S7-1500 product family support MQTT and thereby implement OPC UA PubSub via MQTT.
- Many existing applications and devices already support MQTT and only need to implement the OPC UA protocol mapping "UADP" as a communication partner.

1.2 Principle of Operation

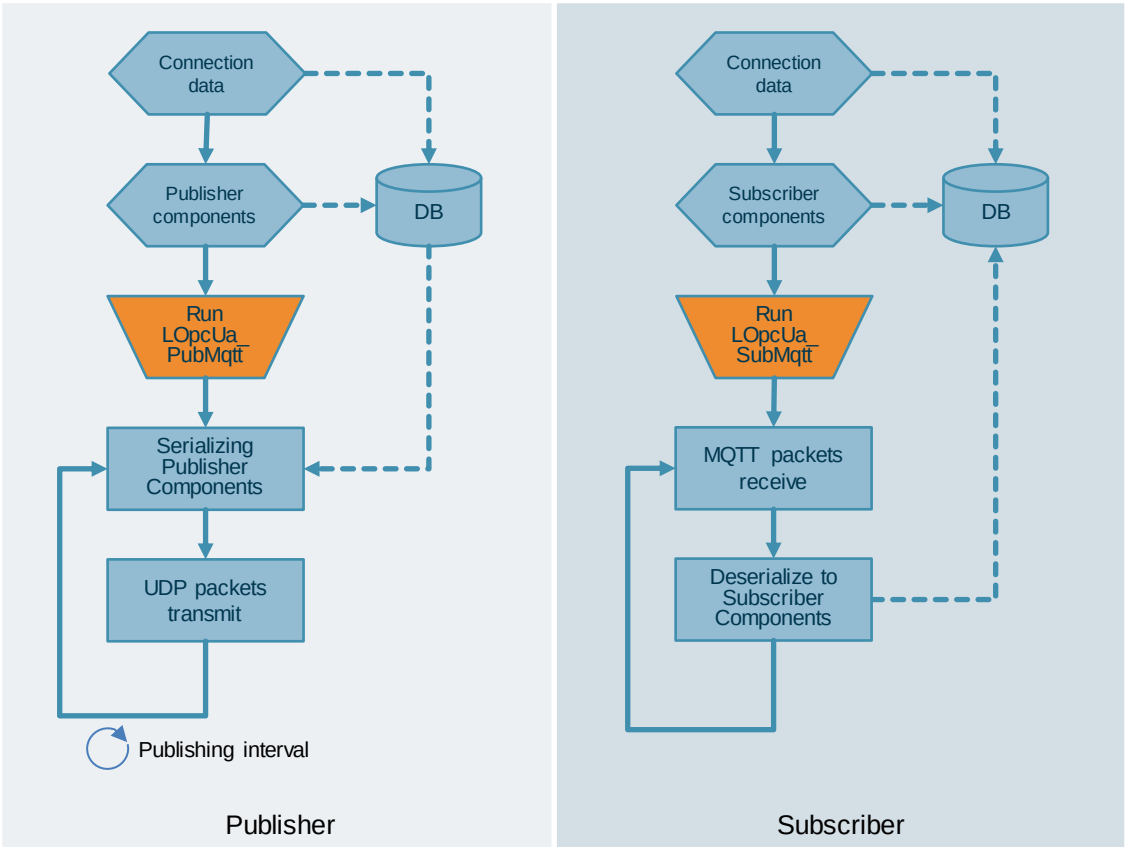
General

PubSub OPC UA applications, with their roles as "Publisher" (sender) and/or "Subscriber" (receiver), are decoupled from each other, which deviates from the Client–Server principle. The number of Subscribers is not important for the Publisher. It is not necessary for Publisher and Subscriber to be directly connected to each other, allowing not only uni-casts but also multi or board casts. OPC UA PubSub specifies how the data must be structured. For more information on structuring and encoding the data, please refer to Section 3 Useful Information.

The Publisher is implemented by the function block "LOpcUa_PubMqtt" and the Subscriber is implemented by the function block "LOpcUa_SubMqtt", which is provided by the library "LOpcUA".

Sequence diagram

Figure 1-2: Functional sequence



Limitations and Restrictions

- The Subscriber block only supports one "ReaderGroup".
- The blocks only support the following data types:
  Bool, SByte, Byte, INT16, UInt16, Int32, UInt32, Int64, UInt64, Float, Double, String, and Guid.

Required knowledge

- Working with TIA Portal
- OPC UA fundamentals and Open User Communication, as well as MQTT

1.3 Components Used

The Application Example has been created with the following hardware and software components:

Table 1-1

Components	Quantity	Article number	Note
SIMATIC STEP 7 Professional V16	1	6ES7822-1AA06-0YA5	TIA Portal V16 or later
SIMATIC S7-1500 CPU 1516F-3 PN/DP	1	6ES7516-3FN01-0AB0	All S7-1500 controllers as of FW 2.0
SIMATIC S7-1500 CPU 1513-1 PN/DP	1	6ES7513-1AL01-0AB0	All S7-1500 controllers as of FW 2.0
Eclipse Mosquitto	1	-	MQTT Broker (\6\)

The listed components can be obtained from the Siemens Industry Mall, for example.

This application example consists of the following components:

Table 1-2

Components	File name	Note
Documentation	109797826_OPC_UA_PubSub_MQTT_DOC_V1_0_de.pdf	This document
Documentation	109780503_Libraries_Comm_Controller_DOC_V1_0_0_de.pdf	Documentation of the block library
Block library	109780503_Libraries_Comm_Controller_LIB_V1_0_0.zip	Block library with PubSub blocks
Sample project	109797826_OPC_UA_PubSub_MQTT_PROJ_V1_0.zip	Sample project for OPC UA PubSub communication based on the PubSub blocks

2 Engineering

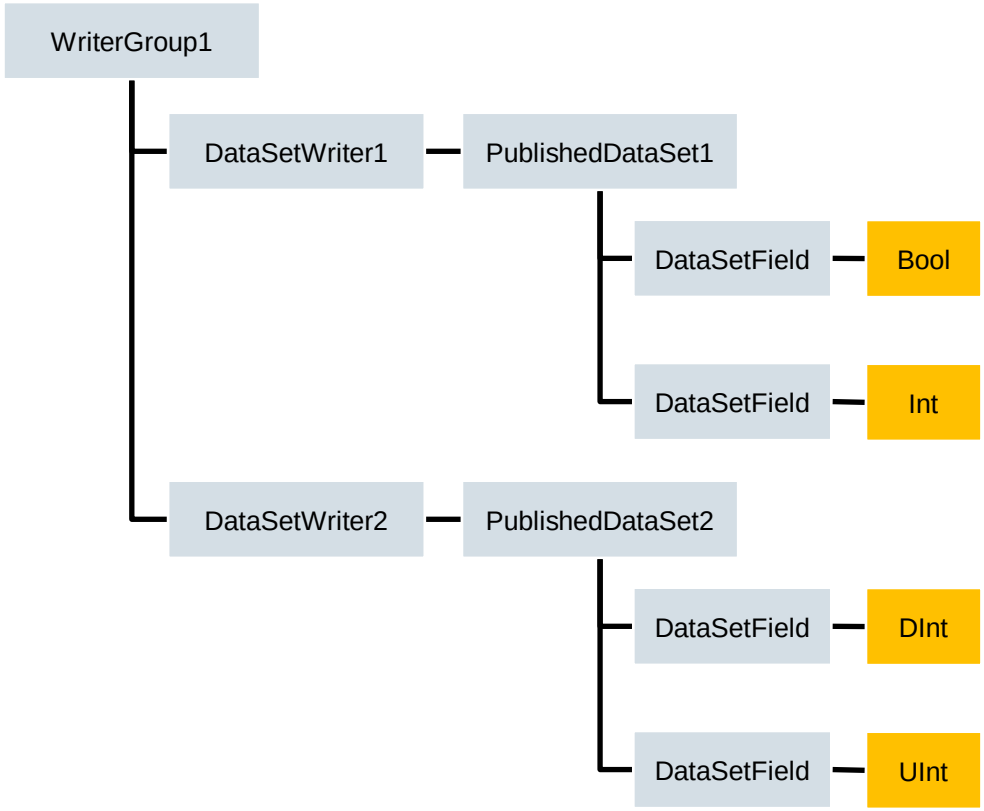
2.1 Project Engineering and Configuration

The following sections explain how to engineer or configure the Publisher and the Subscriber.

Publisher components

The components important for the data exchange are preconfigured in this example and shown in the following figure:

Figure 2-1: Component structure



This division of the PubSub components in this application example results in the data transmission of a "WriterGroup" with two "DataSetWriters", whose "PublishedDataSets" each contain two "DataSetFields". The "DataSetFields" in "PublishedDataSet1" contain a "Bool" and an "UInt". The "DataSetFields" in "PublishedDataSet2" contain a "DInt" and a "UInt".

The value changes of the four transmitted example process tags are simulated by a function, see Section 3.3 Application Example Blocks.

To transfer other data, adjust the PubSub components.

2.1.1 Integrating the Library Blocks

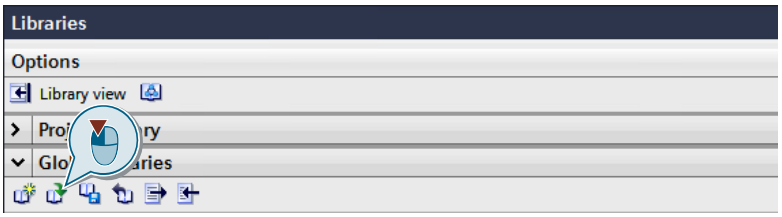
Note

The TIA Portal project provided with this application example is already fully designed and configured and implements the above mentioned Publisher components for you as an example. For the commissioning of the project, you only have to adjust the connection parameters and your PLC type, see chapter 2.2 Commissioning the Application Example.

The following explanations for project engineering and configuration generally apply.

Proceed as follows to include the library blocks in your project:

1. In your TIA Portal project, create one SIMATIC S7-1500 PLC for each Publisher or Subscriber.
2. Download the library "LOpcUa" (\4\).
3. Unzip the zipped file.
4. In TIA Portal, navigate to "Libraries > Global Libraries" and click "Open global library" and select the "al16" file of the extracted ZIP archive.



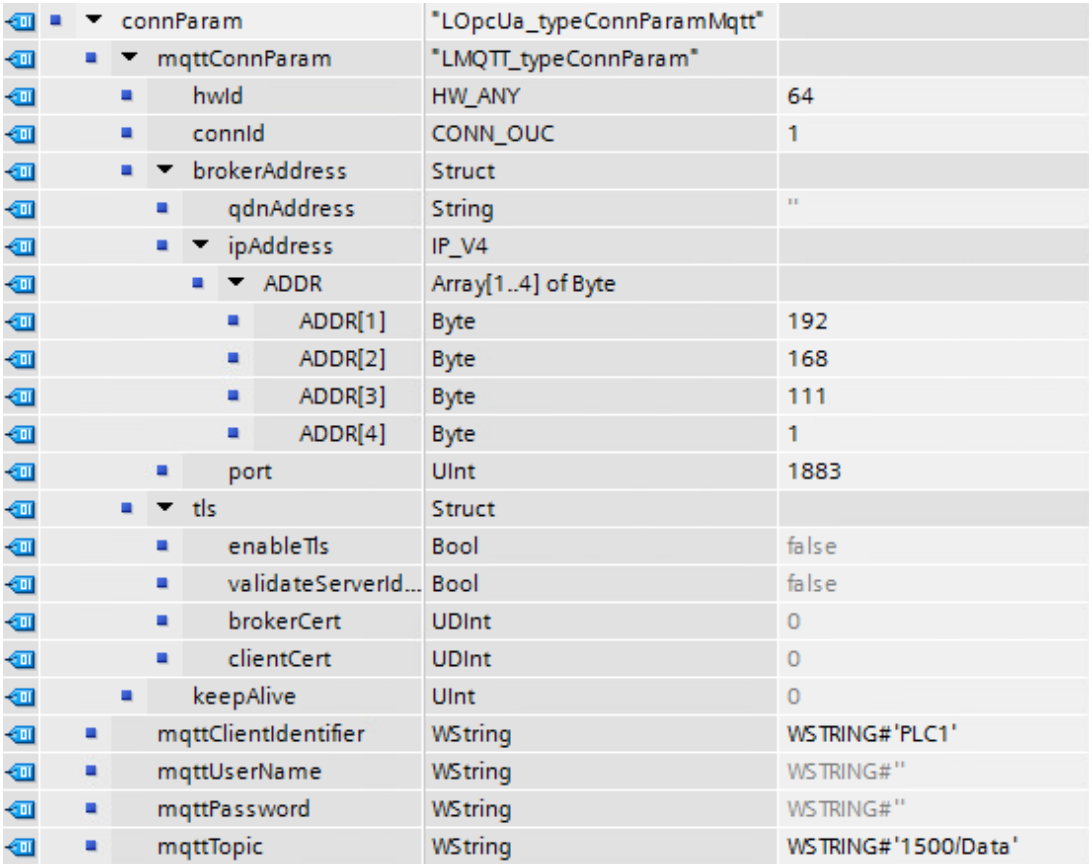
5. In the open library, navigate to "Types > LOpcUa > LOpcUa_MQTT".
   Using drag & drop, move the corresponding function block "LOpcUa_PubMqtt" for the Publisher or "LOpcUa_SubMqtt" for the Subscriber into the "Program blocks" folder. All required data types and functions of the library are then automatically integrated into your project.

2.1.2 Configuration of the Publisher

To configure the Publisher, follow these steps.

Connection parameters

1. Create a tag of the type "LOpcUa_typeConnParamMqtt" in a DB.
2. Fill in the individual fields as follows:



Field	Value	Comment
hwId	0 / 64	"64": Internal Ethernet interface "0": Automatic interface selection
connId	any	Connection ID for the OUC.
qdnAddress	any	Enter the domain name of the MQTT Broker. The domain name must end with a "." If this parameter is used, the IP address entry can be omitted.
ADDR[1-4]	any	IP address of the MQTT Broker. If this parameter is used, the input of the domain name can be omitted.
port	any	Port number of the MQTT Broker
enableTls	TRUE / FALSE	Enter the value "TRUE" if the connection will be secured with TLS.
validateServerIdentity	TRUE / FALSE	Enter the value "TRUE" if the certificate of the MQTT Broker will be validated when establishing the connection.
brokerCert	Any	Certificate ID of the MQTT Broker certificate, from the global certificate manager.
clientCert	any	Certificate ID of the MQTT Client certificate from the global certificate manager.

Field	Value	Comment
keepAlive	any	Enter a value in seconds for the activation of the Keep-Alive mechanism of MQTT. If the tag has the value "0", Keep-Alive is not active.
mqttClientIdentifier	any	Enter the Client identifier that is used when establishing the connection (e.g., "S7-1500")
mqttUserName	any	(Optional) Username for establishing the connection. If no username is entered, the parameter is not evaluated.
mqttPassword	any	(Optional) Password for establishing the connection. If no password is entered, the parameter is not evaluated.
mqttTopic	any	Enter the MQTT topic that will be used in the publish job.

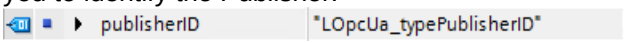
Note

If the MQTT connection is established via the fully qualified domain name, then you must configure a DNS Server in the CPU.

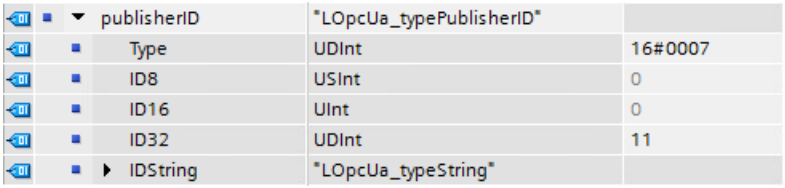
Publisher components

Create a "PublisherID" as follows:

1. Create a tag of type "LOpcUa_typePublisherID" in a DB. This tag maps the "PublisherID" for you to identify the Publisher.



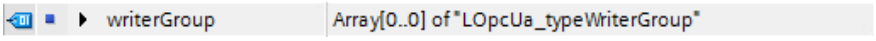
2. Fill in the Publisher ID.



Field	Explanation
Type	Specifies the data type of the Publisher ID to be sent.
IDX	Used field according to the "Type". Get the assignment between "Type" and "ID" from the comments in the data type "LOpcUa_typePublisherID". The corresponding field contains the ID that identifies your Publisher. You can choose the ID freely. Example: *"Type" = "7"* corresponds to *"ID32 (UDInt)"*

Create the "WriterGroups" as follows:

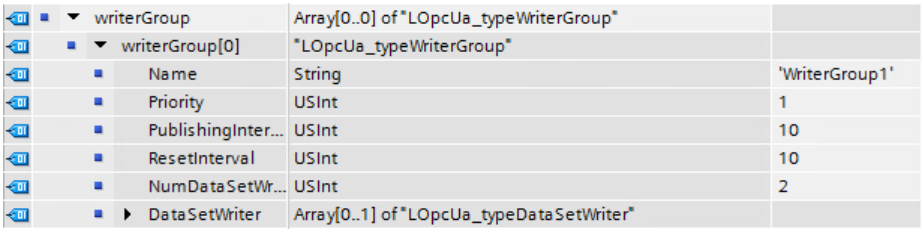
1. Create an array of the type "LOpcUa_typeWriterGroup" in a DB. The size of the array can be freely chosen by you. This array represents the "WriterGroups" and the "DataSetWriter" contained in them.



Note

If you need more "DataSetWriter" than are predefined in the data type "LOpcUa_typeWriterGroup", adjust this data type.

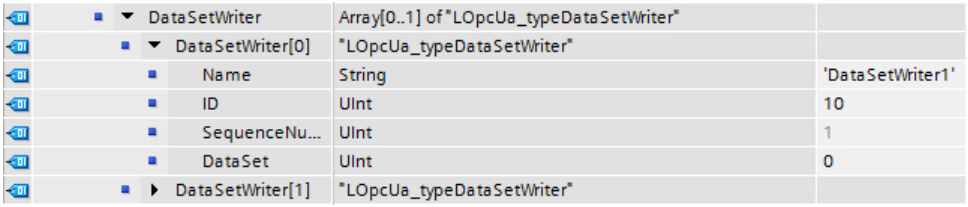
2. Fill in the first index of the "WriterGroups".



writerGroup	Array[0..0] of "LOpcUa_typeWriterGroup"	
writerGroup[0]	"LOpcUa_typeWriterGroup"	
Name	String	'WriterGroup1'
Priority	USInt	1
PublishingInter...	USInt	10
ResetInterval	USInt	10
NumDataSetWr...	USInt	2
DataSetWriter	Array[0..1] of "LOpcUa_typeDataSetWriter"	

Field	Explanation
Name	The name of the "WriterGroup" (optional, will not be transferred).
Priority	The priority for sorting (higher value = higher priority).
PublishingInterval	The publishing interval for the configuration of the send clock. The transmission rate is given by the following formula: *Send clock = OB cycle time * PublishingInterval*
ResetInterval	Must contain the same value as the "PublishingInterval".
NumDataSetWriter	Number of "DataSetWriters" you want to use.

3. Fill in the first index of the "DataSetWriter".



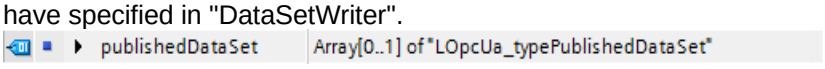
DataSetWriter	Array[0..1] of "LOpcUa_typeDataSetWriter"	
DataSetWriter[0]	"LOpcUa_typeDataSetWriter"	
Name	String	'DataSetWriter1'
ID	UInt	10
SequenceNu...	UInt	1
DataSet	UInt	0
DataSetWriter[1]	"LOpcUa_typeDataSetWriter"	

Field	Explanation
Name	The name of the "DataSetWriter" (optional, will not be transferred).
ID	Identification number of the "DataSetWriter" (any).
SequenceNumber	Is not filled in (required by FB "LOpcUa_PubUdp").
DataSet	Reference to the index of the "PublishedDataSet". Each "DataSet" refers to a different index of the "PublishedDataSets".

4. Repeat step 3 for each additional "DataSetWriter".
5. Repeat steps 2 to 5 for each additional "WriterGroup".

Create the "PublishedDataSets" as follows:

1. Create an array of the type "LOpcUa_typePublishedDataSet" in a DB. The size of the array can be chosen freely. This array maps the "PublishedDataSets" and the "DataSetFields" contained therein. The "DataSetWriters" of the "WriterGroups" each point to a "PublishedDataSet" of this array. You need at least as many "PublishedDataSets" as you have specified in "DataSetWriter".

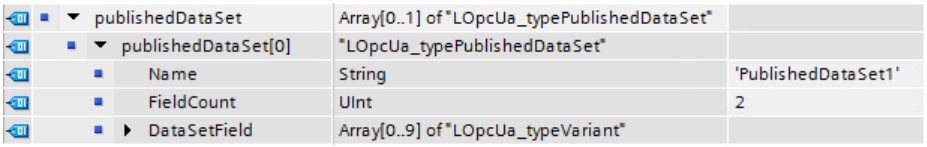


publishedDataSet	Array[0..1] of "LOpcUa_typePublishedDataSet"

Note

If you need more "DataSetFields" than are predefined in the data type "LOpcUa_typePublishedDataSet", adjust this data type.

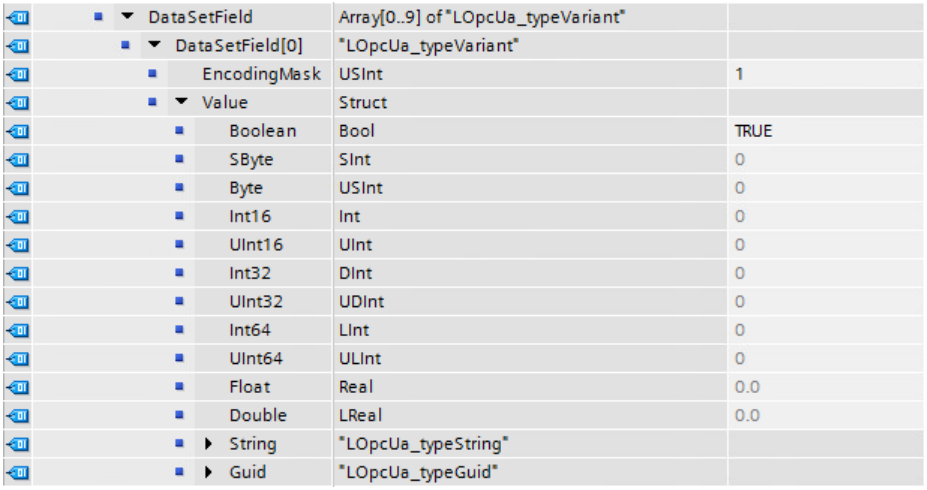
2. Fill in the first index of the "PublishedDataSets".



publishedDataSet	Array[0..1] of "LOpcUa_typePublishedDataSet"	
publishedDataSet[0]	"LOpcUa_typePublishedDataSet"	
Name	String	'PublishedDataSet1'
FieldCount	UInt	2
DataSetField	Array[0..9] of "LOpcUa_typeVariant"	

Field	Explanation
Name	The name of the "PublishedDataSet" (optional, will not be transmitted).
FieldCount	Number of "DataSetFields" in the "PublishedDataSet" to be published.

3. Fill in the first index of the "DataSetFields".



DataSetField	Array[0..9] of "LOpcUa_typeVariant"	
DataSetField[0]	"LOpcUa_typeVariant"	
EncodingMask	USInt	1
Value	Struct	
Boolean	Bool	TRUE
SByte	SInt	0
Byte	USInt	0
Int16	Int	0
UInt16	UInt	0
Int32	DInt	0
UInt32	UDInt	0
Int64	LInt	0
UInt64	ULInt	0
Float	Real	0.0
Double	LReal	0.0
String	"LOpcUa_typeString"	
Guid	"LOpcUa_typeGuid"	

Field	Explanation
EncodingMask	Specifies the data type of the process value to be sent.
Value	Used field according to the "EncodingMask". Take the assignment between "EncodingMask" and "Value" from the comments in the data type "LOpcUa_typeVariant". The corresponding field contains the process value that you can describe in your user program for sending. Example: *"EncodingMask" = "1"* corresponds to *"Boolean"*

4. Repeat step 3 for each required "DataSetField".
5. Repeat steps 2 to 3 for each "PublishedDataSet" you need.

2.1.3 Configuration of the Subscriber

Follow these steps to configure the Subscriber.

Connection parameters

1. Create a tag of the type "LOpcUa_typeConnParamMqtt" in a DB.
2. Fill in the individual fields as follows:

connParam	"LOpcUa_typeConnParamMqtt"	
mqttConnParam	"LMQTT_typeConnParam"	
hwId	HW_ANY	64
connId	CONN_OUC	1
brokerAddress	Struct	
qdnAddress	String	''
ipAddress	IP_V4	
ADDR	Array[1..4] of Byte	
ADDR[1]	Byte	192
ADDR[2]	Byte	168
ADDR[3]	Byte	111
ADDR[4]	Byte	1
port	UInt	1883
tls	Struct	
enableTls	Bool	false
validateServerId...	Bool	false
brokerCert	UDInt	0
clientCert	UDInt	0
keepAlive	UInt	0
mqttClientIdentifier	WString	WSTRING#'PLC1'
mqttUserName	WString	WSTRING#''
mqttPassword	WString	WSTRING#''
mqttTopic	WString	WSTRING#'1500/Data'

Field	Value	Comment
hwId	0 / 64	"64": Internal Ethernet interface "0": Automatic interface selection
connId	any	Connection ID for the OUC.
qdnAddress	any	Enter the domain name of the MQTT Broker. The domain name must end with a "." If this parameter is used, the IP address entry can be omitted.
ADDR[1-4]	any	IP address of the MQTT Broker. If this parameter is used, the input of the domain name can be omitted.
port	any	Port number of the MQTT Broker
enableTls	TRUE / FALSE	Enter the value "TRUE" if the connection will be secured with TLS.
validateServerIdentity	TRUE / FALSE	Enter the value "TRUE" if the certificate of the MQTT Broker will be validated when establishing the connection.
brokerCert	Any	Certificate ID of the MQTT Broker certificate, from the global certificate manager.
clientCert	any	Certificate ID of the MQTT Client certificate from the global certificate manager.

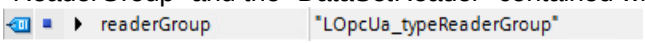
Field	Value	Comment
keepAlive	any	Enter a value in seconds for the activation of the Keep-Alive mechanism of MQTT. If the tag has the value "0", Keep-Alive is not active.
mqttClientIdentifier	any	Enter the Client identifier that is used when establishing the connection (e.g., "S7-1500")
mqttUserName	any	(Optional) Username for establishing the connection. If no username is entered, the parameter is not evaluated.
mqttPassword	any	(Optional) Password for establishing the connection. If no password is entered, the parameter is not evaluated.
mqttTopic	any	Enter the MQTT topic that will be used in the Subscribe job.

Note If the TCP connection will be established via the fully qualified domain name, you must configure a DNS Server in the CPU.

Subscriber Components

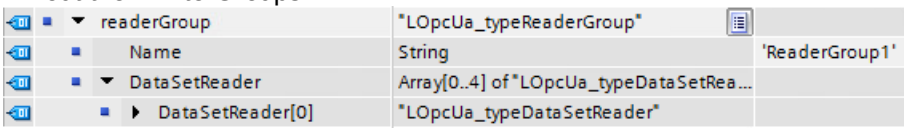
Create the "ReaderGroup" as follows:

1. Create a tag of type "LOpcUa_typeReaderGroup" in a DB. This tag maps the "ReaderGroup" and the "DataSetReader" contained within.



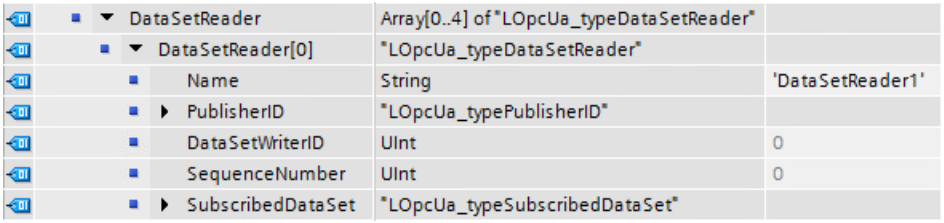
Note If you need more "DataSetReader" than are predefined in the data type "LOpcUa_typeReaderGroup", adjust this data type.

2. Fill out the "WriterGroups".



Field	Explanation
Name	The name of the "ReaderGroup" (optional).

3. Fill in the first index of the "DataSetReader".



Field	Explanation
Name	The name of the "DataSetReader" (optional).

Note If you need more "DataSetFields" than are predefined in the data type "LOpcUa_typeSubscribedDataSet", adjust this data type.

4. Repeat step 3 for each "DataSetReader". The received process values can be taken from the "DataSetFields" in the "DataSetReader" later on.

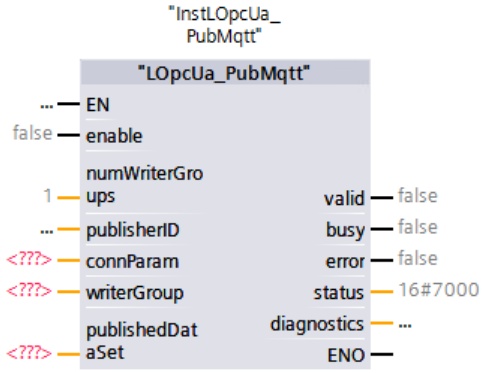
Note Ensure that there are enough "DataSetReader" and "DataSetFields". You need at least as many fields as the Publisher sends.

2.1.4 Calling the PubSub Function Blocks

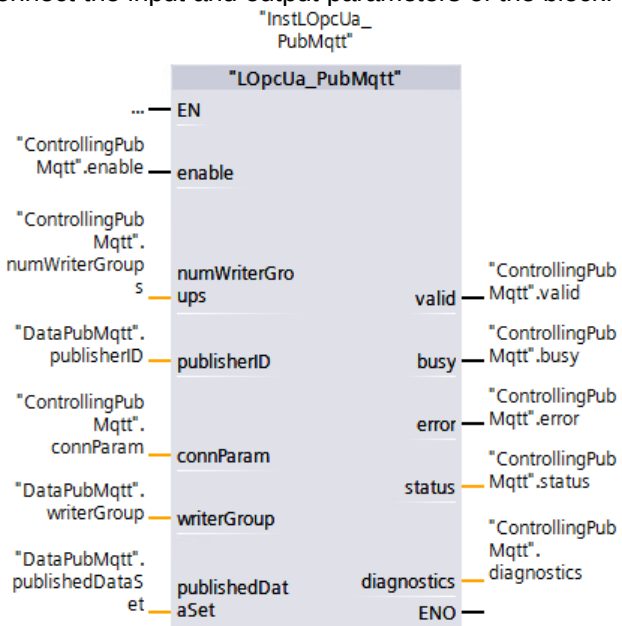
Publisher block

Call up the Publisher block as follows:

1. Call the block "LOpcUa_SubUdp" in a cyclic OB and create an instance data block for it. We recommend an interrupt OB to ensure a deterministic transmit clock.



2. Connect the input and output parameters of the block.



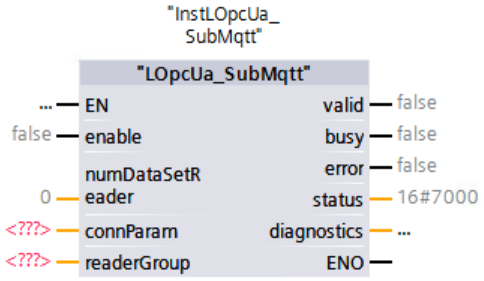
Input and output parameters	Value
enable	Control tag
numWriterGroups	Control tag
publisherID	Predefined Publisher Component "Publisher ID"
connParameter	Predefined connection parameters

Input and output parameters	Value
writerGroup	Predefined Publisher component "WriterGroups"
publishedDataSet	Predefined Publisher Component "PublishedDataSet"
valid	Diagnostics output
busy	Diagnostics output
error	Diagnostics output
status	Diagnostics output
diagnostics	Diagnostics output

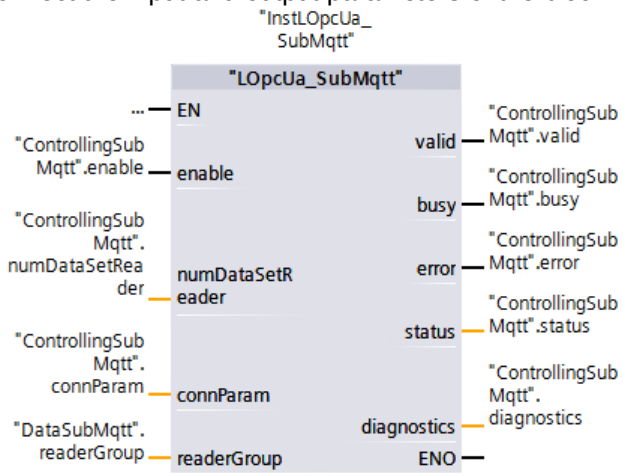
For the meaning of the control tags and diagnostic outputs, refer to the documentation for the block library (\4\).

Subscriber block

1. Call the block "LOpcUa_SubUdp" in a cyclic OB and create an instance data block for it. We recommend the OB1 cycle for this purpose so that the received data can be processed with minimum delay.



2. Connect the input and output parameters of the block.



Input and output parameters	Value
enable	Control tag
numDataSetReader	Control tag
connParam	Predefined connection parameters
readerGroup	Predefined Subscriber component "ReaderGroup"
valid	Diagnostics output
busy	Diagnostics output
error	Diagnostics output
status	Diagnostics output
diagnostics	Diagnostics output

For the meaning of the control tags and diagnostic outputs, refer to the documentation for the block library (\4\).

2.2 Commissioning the Application Example

Publisher and Subscriber

Put the application example into operation as follows:

1. Download the sample project "109782455_OPC_UA_PubSub_MQTT_PROJ_V1_0.zip" (\2\).
2. Unzip the zipped file.
3. Double-click to open the project file "OPCUA_PubSub_MQTT.ap16".
4. (Optional) Change the PLC type to match your existing PLCs. To do this, right-click the PLCs in the project directory and click "Change device" and follow the instructions.
5. (Optional) Adjust the connection parameters of the Publisher and Subscriber if necessary. You will find the connection parameters in the following places:
   - "Publisher_PLC": Data block "ControllingPub", see Section 2.1.2 Configuration of the Publisher.
   - "Subscriber_PLC": Data block "ControllingSub", see Section 2.1.3 Configuration of the Subscriber.
6. Compile and load the hardware and software of the "Publisher_PLC" as well as the "Subscriber_PLC" into the corresponding PLCs.

Detailed information on the user program of the sample project can be found Section 3.3 Application Example Blocks.

MQTT Broker

Start the MQTT Broker as follows (for Windows):

1. Download the Windows installer (\6\).
2. Install "Mosquitto" by double-clicking the EXE file.
3. Create a text file.
4. Enter the following parameters in the file:

```
allow_anonymous true
listener 1883 0.0.0.0
```

   With "allow_anonymous", you allow the MQTT Clients to connect to the Broker without login data (username/password). With "listener", you enable a TCP listener that allows not only local connections ("127.0.0.1"). Here, "0.0.0.0" is a wildcard, so that the Broker can be reached via all IP addresses of the host system. Alternatively, you can also enter a static IP address of one of your interfaces.

5. Save the configuration file under "C:\Program Files\Mosquitto".
6. Change the name and the ending of the file to "mosquitto.conf".
7. After installation, the Broker starts automatically with Windows. To start the service manually, open the command prompt as Administrator and enter the following command:

```
net start mosquitto
```

   To stop the service manually, enter the following command:

```
net stop mosquitto
```

2.3 Operation of the Application Example

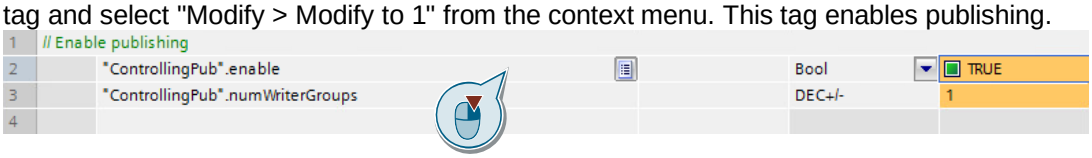
The application example is controlled via Observation Tables. The following steps explain how to use the example:

Publisher

Control the Publisher as follows:

1. Navigate in the project directory of the "Publisher_PLC" to "Watch and force tables" and double-click to open the Observation Table "Controlling".
2. Click "Monitor all" in the Observation Table to monitor the preconfigured tags.

3. Adjust the "numWriterGroups" tag if you have changed the Publisher configuration.
   Set the tag "enable" in the "Enable publishing" section to "TRUE". To do this, right-click the tag and select "Modify > Modify to 1" from the context menu. This tag enables publishing.



1	// Enable publishing			
2	"ControllingPub".enable		Bool	TRUE
3	"ControllingPub".numWriterGroups		DEC+/-	1
4				

4. Observe the tags "valid" and "error" in the "Diagnostic publishing" section. If "error" is set, check the connection parameters and the configuration of the Publisher components and repeat the procedure.

// Diagnose publishing			
"ControllingPub".valid		Bool	TRUE
"ControllingPub".busy		Bool	TRUE
"ControllingPub".error		Bool	FALSE
"ControllingPub".status		Hex	16#7002
"ControllingPub".diagnostics.stateNumber		DEC+/-	0
"ControllingPub".diagnostics.status		Hex	16#0000
"ControllingPub".diagnostics.subfunctionStatus		Hex	16#0000

5. As long as the tag "enable" remains set to "TRUE", the Publisher sends the "DataSetFields" to all Subscribers at the specified send interval (in the example: 1000 ms).
6. Set the "enable" tag to "FALSE" to stop publishing.

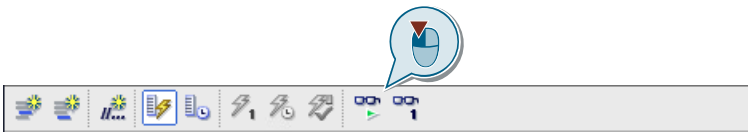
The "Data and Publisher ID to be published" section shows you the simulated process values and the predefined Publisher ID.

// Data and publisher ID to be published			
"PubUDP_Data".publisherID.ID32		DEC	11
"PubUDP_Data".publishedDataSet[0].DataSetField[0].Value.Boolean		Bool	TRUE
"PubUDP_Data".publishedDataSet[0].DataSetField[1].Value.Int16		DEC+/-	11203
"PubUDP_Data".publishedDataSet[1].DataSetField[0].Value.Int32		DEC+/-	813_995_528
"PubUDP_Data".publishedDataSet[1].DataSetField[1].Value.UInt16		DEC	55254

Subscriber

You control the Subscriber as follows:

1. Navigate in the project directory of the "Subscriber_PLC" to "Watch and force tables" and double-click to open the Observation Table "Controlling".
2. Click "Monitor all" in the Observation Table to monitor the preconfigured tags.



3. Adjust the "numDataSetReader" tag if you have changed the Publisher configuration.
   Set the tag "enable" in the section "Enable subscribing" to "TRUE". To do this, right-click the tag and select "Modify > Modify to 1" from the context menu. This tag enables subscribing.

// Enable subscribing			
"ControllingSub".enable		Bool	TRUE
"ControllingSub".numDataSetReader		DEC+/-	2

4. Observe the tags "valid" and "error" in the "Diagnostic subscribing" section. If "error" is set, check the connection parameters and the configuration of the Subscriber components and repeat the process.

// Diagnose subscribing			
"ControllingSub".valid		Bool	TRUE
"ControllingSub".busy		Bool	TRUE
"ControllingSub".error		Bool	FALSE
"ControllingSub".status		Hex	16#7002
"ControllingSub".diagnostics.stateNumber		DEC+/-	0
"ControllingSub".diagnostics.status		Hex	16#0000
"ControllingSub".diagnostics.subfunctionStatus		Hex	16#0000

5. As long as the "enable" tag remains "TRUE", the Subscriber receives the Publisher's "DataSetFields".
6. Set the tag "enable" to "FALSE" to stop subscribing.

In the "Received Publisher ID" section, the received Publisher IDs of the "DataSetWriter" are displayed.

// Received publisher ID			
"SubUDP_Data".readerGroup.DataSetReader[0].PublisherID.ID32		DEC	11
"SubUDP_Data".readerGroup.DataSetReader[1].PublisherID.ID32		DEC	11

In the section "Received process data", the received process values from the "DataSetFields" are displayed.

// Received process data			
"SubUDP_Data".readerGroup.DataSetReader[0].SubscribedDataSet.Dat		Bool	FALSE
"SubUDP_Data".readerGroup.DataSetReader[0].SubscribedDataSet.Dat		DEC+/-	22859
"SubUDP_Data".readerGroup.DataSetReader[1].SubscribedDataSet.Dat		DEC+/-	-722_183_893
"SubUDP_Data".readerGroup.DataSetReader[1].SubscribedDataSet.Dat		DEC	43598

3 Useful Information

3.1 Fundamentals

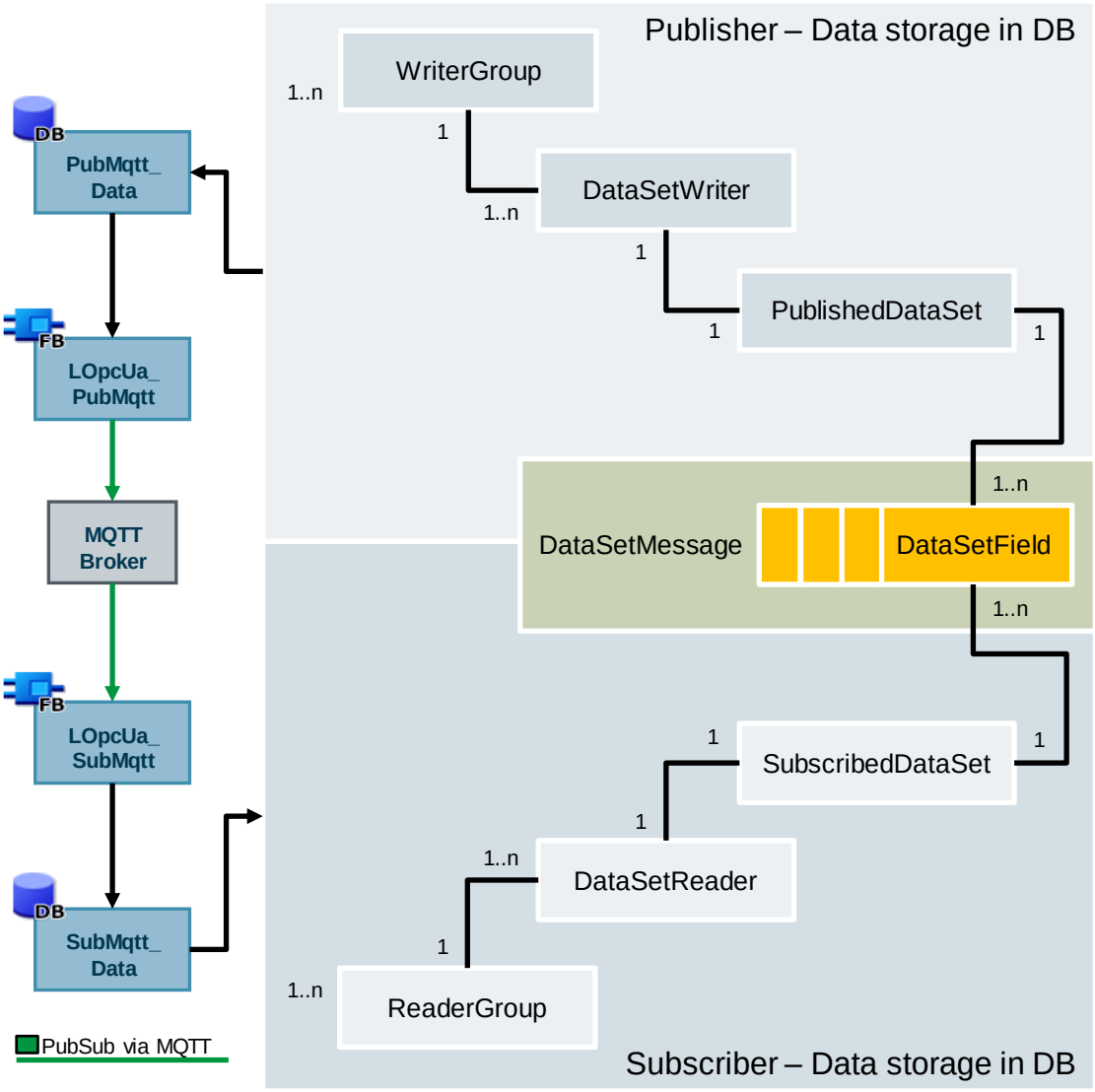
Fundamentals of OPC UA PubSub can be found in the OPC UA specification (OPC 10000-14) (\3\).

3.2 Functionality Details

3.2.1 Schematic Representation

The following figure shows the most important relationships between the components involved in sending data from a Publisher to a Subscriber:

Figure 3-1



To map one or more "WriterGroups" or "ReaderGroups", both the Publisher and the Subscriber function blocks require a data block in which the PubSub data (metadata and process data) are provided according to the schema shown above.

For the implementation of the individual components in data blocks, the "LOpcUa" library provides suitable PLC data types that map the required structures.

The individual components give you the possibility to structure your PubSub data individually. For example, you can publish four process values via a "WriterGroup" divided into four "DataSetWriters". You can also publish four tags via a "WriterGroup" and a "DataSetWriter".

If a component is required more than once ("1..n"), it must be created as an array in the data block.

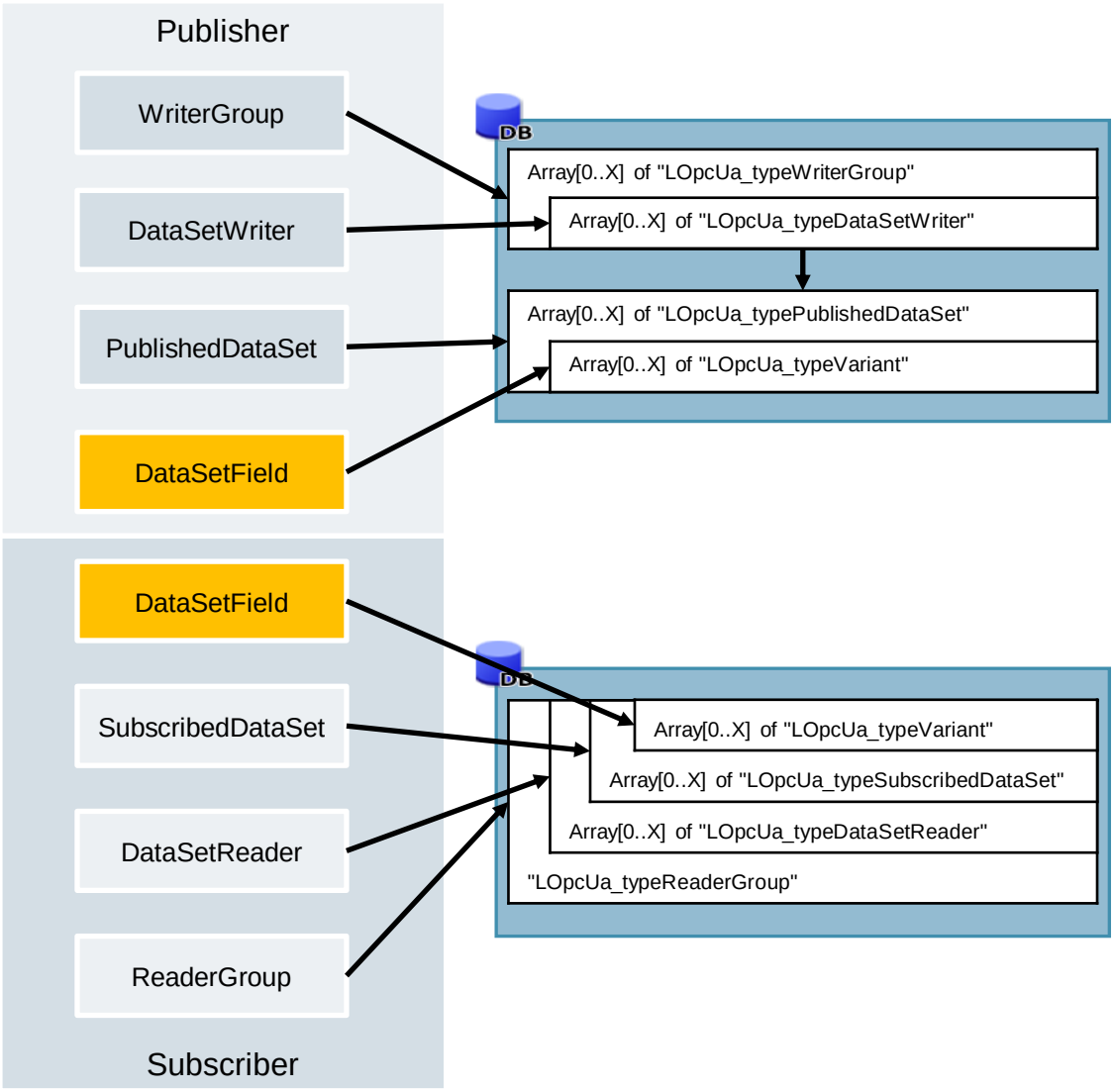
The "LOpcUa_PubMqtt" block accesses the "WriteGroups" and serializes the "DataSetMessage" from the data contained within. Subsequently, it transmits ("Publish") these to the Broker with the help of the library module "LMQTT_Client".

The block "LOpcUa_SubMqtt" receives the "DataSetMessage" via the library block "LMQTT_Client" from the Broker and deserializes the data to the data structure of the "ReaderGroup" created.

3.2.2 Data Structure of the PubSub Components in the Data Block

The following figure describes how the components of the PubSub communication are mapped in data blocks:

Figure 3-2



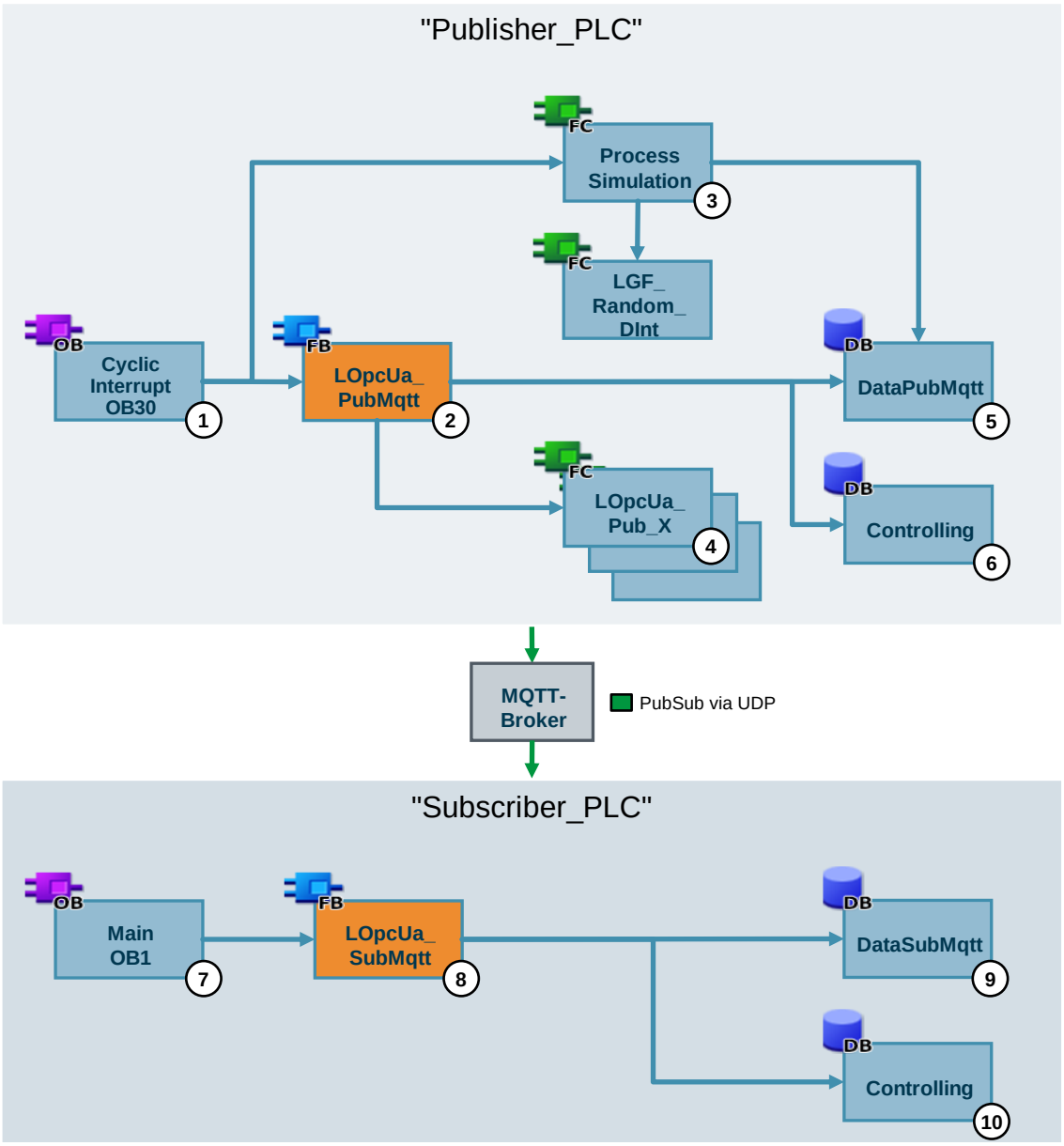
All required data types can be found in the library "LOpcUa" (\4\).

3.3 Application Example Blocks

The application example consists of a TIA Portal project and contains two PLCs. One PLC assumes the role of Publisher and the other PLC that of Subscriber.

The following figure describes the structure, the blocks, and their call hierarchy in the application example:

Figure 3-3



The following table explains the individual blocks from Figure 3-3:

Table 3-1

no.	Description
1.	Interrupt OB for calling the Publisher block "LOpcUa_PubMqtt". In this application example, the interrupt OB is used to ensure deterministic send cycles.
2.	Publisher block "LOpcUa_PubMqtt" for sending process data.
3.	Function for generating process data to be sent by the Publisher block. The function writes the process data into the data block "DataPubMqtt". Internally, the library block "LGF_Random_DInt" (\5\) is used to generate random numbers.
4.	Functions for serialization (end coding) of process data. The Publisher block requires these functions to prepare the PubSub communication.
5.	Data block for mapping the Publisher components. The data block contains the process and metadata of the PubSub communication.
6.	Data block for controlling the Publisher module. All Publisher control tags are contained in this block.
7.	Cyclic OB for calling the Subscriber block "LOpcUa_SubMqtt". In this application example, cyclic OB 1 is used to obtain PubSub data with minimal delay.
8.	Subscriber block "LOpcUa_SubMqtt" for receiving process data.
9.	Data block for mapping the Subscriber components. The data block contains the received process data of the PubSub communication.
10.	Data block for controlling the Subscriber block. All control tags of the Subscriber are contained in this block.

4 Appendix

4.1 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:

support.industry.siemens.com/cs/my/src

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

4.2 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

4.3 Links and literature

Table 4-1

no.	Subject
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/109797826
\3\	Link to the specification of PubSub (OPC 10000-14) https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub/
\4\	Link to the block library https://support.industry.siemens.com/cs/ww/en/view/109780503
\5\	Library of General Functions (LGF) https://support.industry.siemens.com/cs/ww/en/view/109479728
\6\	Eclipse Mosquitto™ (Open Source MQTT Broker) https://mosquitto.org/

4.4 Change documentation

Table 4-2

Version	Date	Change
V1.0	11/2021	First version