



**SIEMENS**



Industry Online Support

Home

# Libraries for Communication for SIMATIC Controllers

SIMATIC Controllers, TIA Portal

<https://support.industry.siemens.com/cs/ww/en/view/109780503>

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

Legal information .....	2
<b>1 Introduction .....</b>	<b>6</b>
<b>2 LCom .....</b>	<b>8</b>
2.1 Overview .....	8
2.1.1 Range of Functions .....	8
2.1.2 Components of the Library.....	11
2.1.3 Validity .....	11
2.2 LCom_Communication .....	12
2.3 PLC Data Types.....	14
2.4 Integration into the User Project.....	18
2.5 Error Handling .....	18
2.5.1 Status Outputs.....	18
2.5.2 Status Messages.....	20
<b>3 LFTP.....</b>	<b>29</b>
3.1 Overview .....	29
3.1.1 Range of Functions .....	29
3.1.2 Components of the Library.....	30
3.1.3 Validity .....	30
3.2 LFTP_Client .....	31
3.2.1 Interface Description.....	31
3.2.2 Error Handling.....	32
3.3 PLC Data Types.....	33
3.4 Integration into the user project.....	34
<b>4 LHTTP .....</b>	<b>35</b>
4.1 Overview .....	35
4.1.1 Range of Functions .....	35
4.1.2 Components of the Library.....	36
4.1.3 Validity .....	36
4.2 LHTTP_Get.....	37
4.2.1 Interface Description.....	37
4.2.2 Operation .....	38
4.3 LHTTP_PostPut .....	40
4.3.1 Interface Description.....	40
4.3.2 Operation .....	41
4.4 LHTTP_FindStringInArray.....	43
4.5 LHTTP_ExtractStringFromArray .....	43
4.6 LHTTP_ExtractStringFromArrayExt .....	44
4.7 PLC Data Types.....	45
4.8 Error Handling .....	45
<b>5 LMQTT .....</b>	<b>48</b>
5.1 Overview .....	48
5.1.1 Range of Functions .....	48
5.1.2 Components of the Library.....	48
5.1.3 Validity .....	48

## Table of Contents

5.2	LMQTT_Client.....	49
5.2.1	Interface Description.....	49
5.2.2	Error Handling.....	52
5.2.2.1	Block Outputs without Error.....	52
5.2.2.2	Block Outputs in Case of Error.....	53
5.3	LMQTT_ConvertToUtf8.....	55
5.4	PLC Data Types.....	55
5.5	Integration into the User Project.....	56
<b>6</b>	<b>LMindConn.....</b>	<b>57</b>
6.1	Overview.....	57
6.1.1	Range of Functions.....	57
6.1.2	Components of the Library.....	57
6.1.3	Validity.....	58
6.2	LMindConn_MQTT.....	58
6.2.1	Interface Description.....	58
6.2.2	Operation.....	60
6.2.3	Error Handling.....	63
6.3	LMindConn_ConvertDatapoint.....	64
6.4	LMindConn_ConvertDatapoints50.....	65
6.5	PLC Data Types.....	66
6.6	Integration into the User Project.....	67
<b>7</b>	<b>LOpcUa.....</b>	<b>68</b>
7.1	Overview.....	68
7.1.1	Range of Functions.....	68
7.1.2	Components of the Library.....	69
7.1.3	Validity.....	70
7.2	PubSub via UDP.....	71
7.2.1	Principle of Operation.....	71
7.2.2	LOpcUa_PubUdp.....	73
7.2.3	LOpcUa_SubUdp.....	74
7.2.4	Error Handling.....	75
7.2.5	Integration into the User Project.....	75
7.3	PubSub via MQTT.....	76
7.3.1	Principle of Operation.....	76
7.3.2	LOpcUa_PubMqtt.....	78
7.3.3	LOpcUa_SubMqtt.....	79
7.3.4	LOpcUa_PubMqttJson.....	81
7.3.5	LOpcUa_SubMqttJson.....	82
7.3.6	Error Handling.....	83
7.3.7	Integration into the User Project.....	83
7.4	PLC Data Types.....	84
<b>8</b>	<b>LSNMP.....</b>	<b>89</b>
8.1	Overview.....	89
8.1.1	Range of Functions.....	89
8.1.2	Components of the Library.....	91
8.1.3	Validity.....	91
8.2	LSNMP_Get.....	92
8.3	LSNMP_GetBulk.....	93
8.4	LSNMP_Set.....	95

## Table of Contents

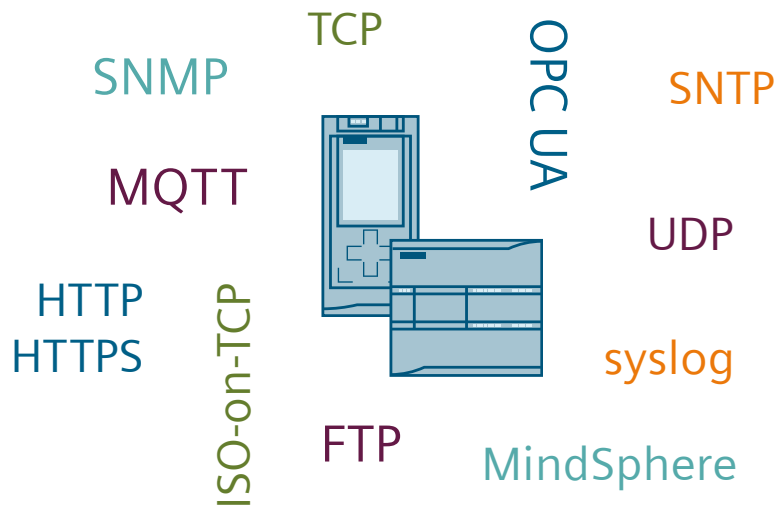
8.5	LSNMP_SendTrap .....	96
8.6	PLC Data Types .....	97
8.7	Integration into the User Project.....	98
8.8	Error Handling .....	99
<b>9</b>	<b>LSNTP .....</b>	<b>100</b>
9.1	Overview .....	100
9.1.1	Range of Functions .....	100
9.1.2	Components of the Library.....	101
9.1.3	Validity .....	101
9.2	LSNTP_Server .....	102
9.2.1	Interface Description.....	102
9.2.2	Error Handling .....	103
<b>10</b>	<b>LSyslog.....</b>	<b>104</b>
10.1	Overview .....	104
10.1.1	Range of Functions .....	104
10.1.2	Components of the Library.....	105
10.1.3	Validity .....	105
10.2	LSyslog_Send .....	106
10.2.1	Interface Description.....	106
10.2.2	Error Handling .....	107
10.3	PLC Data Types .....	108
10.4	Integration into the User Project.....	108
<b>11</b>	<b>Master Copies.....</b>	<b>109</b>
11.1	OUC Master Copies.....	109
11.1.1	Overview .....	109
11.1.1.1	Range of Functions .....	109
11.1.1.2	Components .....	109
11.1.1.3	Validity .....	109
11.1.2	IsoOnTcpTemplate .....	110
11.1.3	TcpTemplate .....	112
11.1.4	TcpSecTemplate .....	114
11.1.5	UdpTemplate.....	116
11.1.6	Error Handling .....	118
11.2	OPC UA structured data types.....	118
<b>12</b>	<b>Useful Information.....</b>	<b>119</b>
12.1	Libraries in TIA Portal .....	119
12.2	Diagnostics.....	120
<b>13</b>	<b>Appendix .....</b>	<b>122</b>
13.1	Service and Support.....	122
13.2	Industry Mall.....	123
13.3	Links and Literature .....	123
13.4	Change Documentation .....	124

# 1 Introduction

## Overview

The Libraries for Communication are a collection of blocks for various communication tasks, functions, and protocols for SIMATIC Controllers.

Figure 1-1



## Range of Functions

The following libraries are included:

- **LCom:**  
This library enables communication based on TCP and provides additional communication functionalities using its own protocol (see Section [2](#)).
- **LFTP:**  
With this library, a controller can act as an FTP client (see Section [3](#)).
- **LHTTP:**  
This library enables data exchange with a web server in the local network or on the internet via HTTP or HTTPS (see Section [3](#)).
- **LMQTT:**  
This library enables the communication of a controller as MQTT Client (see Section [5](#)).
- **LMindConn:**  
This library enables the direct connection of a controller to MindSphere (see Section [6](#)).
- **LOpcUa:**  
This library provides function blocks for OPC UA PubSub communication (see Section [7](#)).
- **LSNMP:**  
This library can be used to monitor and control SNMP-enabled network components from a controller or to send messages to a network management system (see Section [8](#)).
- **LSNTP:**  
With this library, a controller can act as an SNTTP server to synchronize the time throughout different areas of the system (see Section [9](#)).
- **LSyslog:**  
This library allows sending syslog messages to a syslog sever over UDP or TCP with optional TLS encryption (see Section [10](#)).

In addition, the library provides master copies that you can use to easily implement your own communication functions:

- Master copies for communication via TCP, UDP, and ISO-on-TCP (see Section [11.1](#))
- Master copies for OPC UA methods
- Master copies for common OPC UA structured data types
- Constants for OPC UA status codes

### Application

These libraries are available for TIA Portal V16 and TIA Portal V17.

All included libraries are valid for SIMATIC S7-1500 and S7-1200 controllers. If the respective library is also valid for other controllers, this is described in the section for the corresponding library.

## 2 LCom

### 2.1 Overview

#### 2.1.1 Range of Functions

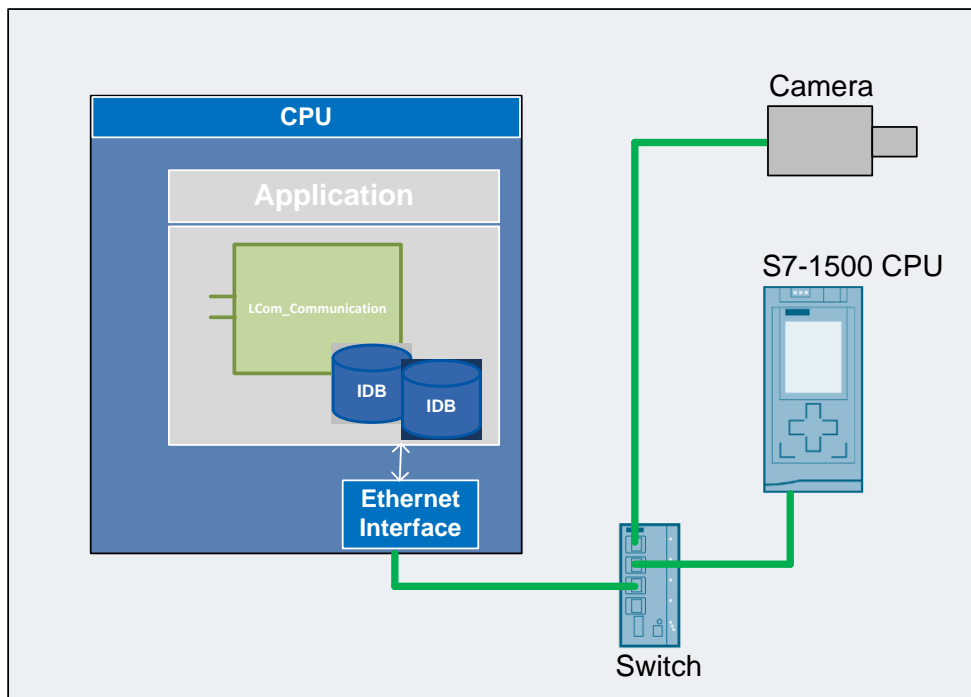
##### Possible applications for the use of the LCom library

The "LCom\_Communication" function block of the LCom library is used to establish a point-to-point full duplex connection via Industrial Ethernet, based on the TCP standard.

The function block can be used for standard TCP communication to other devices (e.g. camera, controller).

Since the range of functions of the TCP transport protocol is not sufficient for many applications in the automation sector, a separate protocol (LCom protocol) has been defined and implemented in the LCom block library. The LCom protocol enables additional communication functionalities.

Figure 2-1: Application scenarios



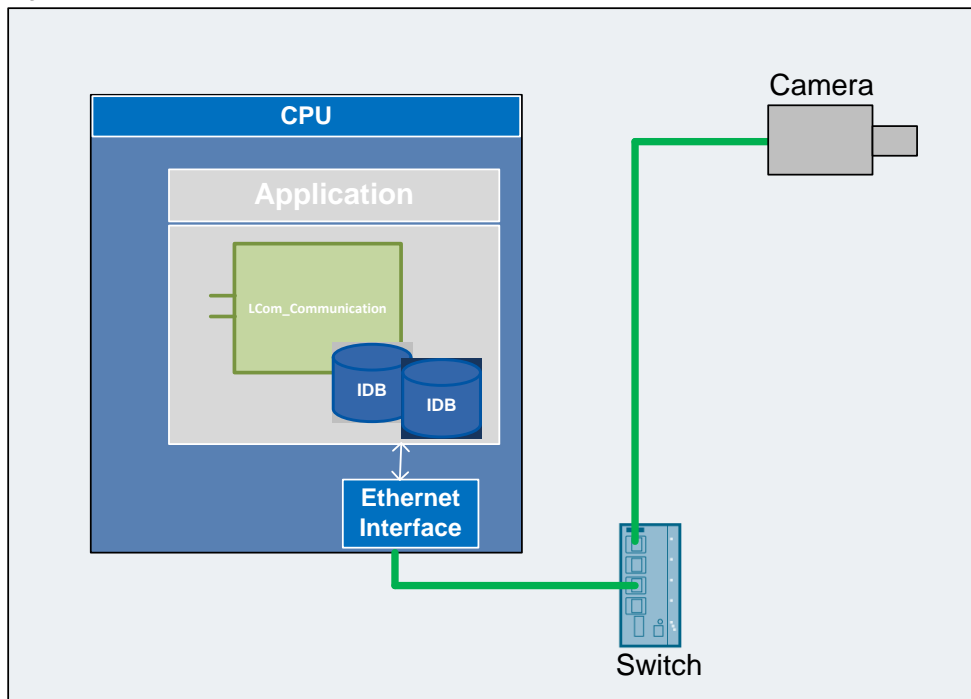


Some scenarios for a possible use of the LCom library are shown below:

### Scenario 1

The "LCom\_Communication" function block is used for standard TCP communication. The TCP transport protocol ensures that a continuous flow of data is transferred. TCP is not packet-oriented and, therefore, does not permit the transmission of data records with a defined overall length.

Figure 2-2: LCom\_Communication – Standard TCP communication



The LCom library offers the following advantages:

- The user does not need to know and program the OUC system function to establish, send/receive, and terminate a connection
- In the event of errors/faults, the function block automatically closes the connection and attempts to reestablish it
- Cyclic data transmission, the user defines the cycle time
- One time data transmission
- Response from sender/receiver about successfully transmitted data at application level
- The function block contains a data structure for diagnostics

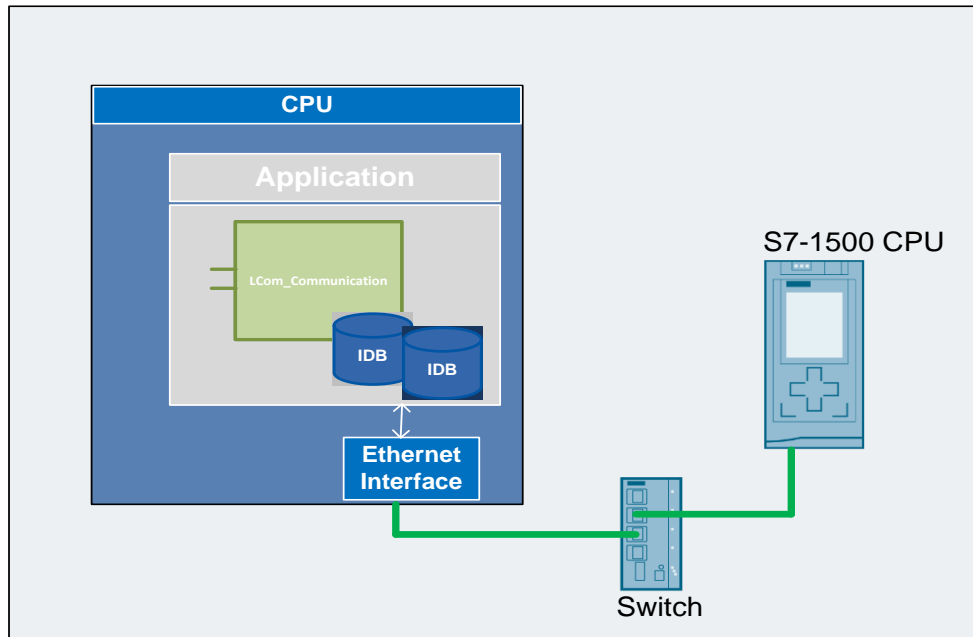
## Scenario 2

When using the LCom protocol, data records of a defined total length can be sent and received consistently by the partner. In order to quickly detect a connection failure, cyclic vital signs are sent. The cycle time of the periodic vital signs is specified by the user.

To synchronize the time of two controllers, the current time of one controller can be sent to the partner and transferred there as the system time.

To use the additional communication functionalities, the communication partner (e.g. S7-1500 CPU) must support the LCom protocol.

Figure 2-3: LCom\_Communication – LCom protocol



The LCom library offers the following advantages:

- The user does not need to know and program the OUC system function to establish, send/receive, and terminate a connection
- In the event of errors/faults, the function block automatically closes the connection and attempts to reestablish it
- Cyclic data transmission, the user defines the cycle time
- One time data transmission
- Consistent data transmission
- Synchronizing the communication parameters
- Monitoring of the communication link by cyclically sending vital signs (fast response times in case of link failure). For pure TCP communication, this is typically in the range of seconds.
- Data records with defined length up to 64 kB with LCom protocol V1
- Data records with defined length up to 16 MB with LCom protocol V2
- Response from sender/receiver about successfully transmitted data at application level
- Simple time synchronization
- The function block contains a data structure for diagnostics

## 2.1.2 Components of the Library

The "LCom" library contains the following objects.

### Function blocks

Table 2-1: Function blocks of the library

Name	Version	Description
LCom_Communication	V2.1.0	Controls the communication between the controller and the partner.

### PLC data types

Table 2-2: PLC library data types

Name	Version	Description
LCom_typeConfig	V2.0.1	Contains the necessary communication parameters.
LCom_typeDiagnostics	V2.0.1	Provides a detailed diagnostic structure.

### Master copies

An example how a configuration and execution of a communication with user defined data in connection with the "LCom\_Communication" FB looks like is provided in the master copies templates.

In addition, constants are provided for global usage. With help of that for example, the status and error codes of "LCom\_Communication" FB can be processed more readable in user application program.

Table 2-3 Master copies templates

Name	Type	Description
LCom_Example	OB	Implementation of an example communication with LCom
LCom_ExampleComData	DB	Data of example communication
LCom_ExampleComBuffers	DB	Send- and receive buffer
LCom_typeComUserData	Data type	User defined data type of example communication
LCom_Constants	PLC tags	LCom constants (i.e. status and error codes) for global usage in user application program

## 2.1.3 Validity

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.4
- CP 1543-1, CP 1542SP-1, CP 1542SP-1 IRC, CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V2.0

## 2.2 LCom\_Communication

### Parameters

Figure 2-4: LCom\_Communication

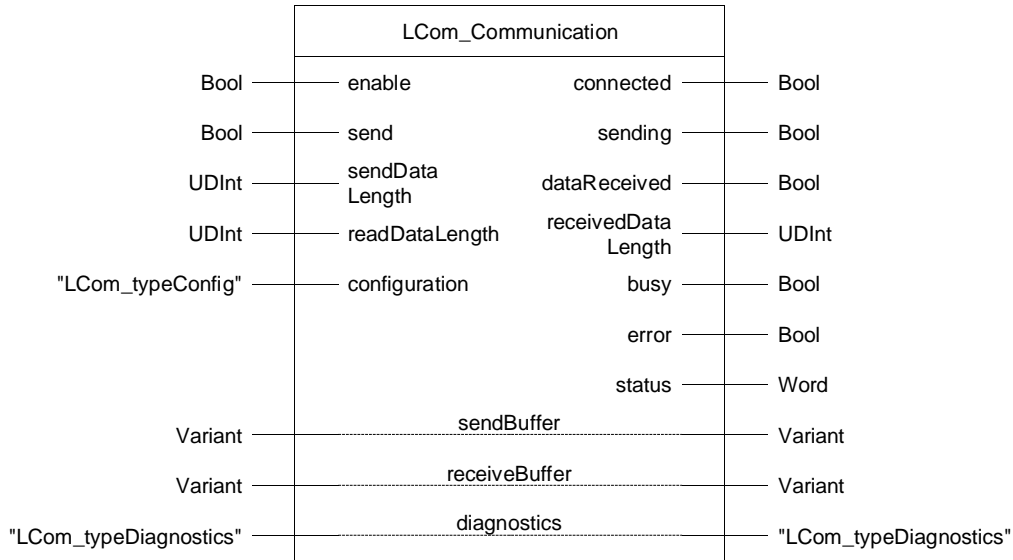


Table 2-4: Parameters of LCom\_Communication

Name	Declaration	Data type	Comment
enable	Input	Bool	<ul style="list-style-type: none"> <li>TRUE: Starts the FB with a rising edge. The connection to the partner is established automatically.</li> <li>FALSE (default): With a falling edge, the connection is broken.</li> </ul>
send	Input	Bool	<ul style="list-style-type: none"> <li>TRUE: Sends the data connected to the "sendBuffer" input parameter.</li> <li>FALSE (default): Data that is connected at the "sendBuffer" input parameter is not sent.</li> </ul>
sendDataLength	Input	UDInt	Data length to be sent in bytes. (default: 4294967295)
readDataLength	Input	UDInt	Without LCom protocol, ("configuration.connection.comService" = "LCOM_TCP_CONNECTION"): <ul style="list-style-type: none"> <li>0: Data from the TCP buffer is not read. FB, therefore, does not receive any data from the partner.</li> <li>1..4294967294: Number of bytes that must be received by the interface before the output tag "dataReceived" = TRUE.</li> <li>4294967295, 16#FFFFFF (default): All data available at the interface is read (ad hoc mode).</li> </ul>

## 2 LCom

Name	Declaration	Data type	Comment
			<p>With LCom protocol, ("configuration.connection.comService" = "LCOM_LCOM_CONNECTION");</p> <ul style="list-style-type: none"> <li>0: Data from the TCP buffer is not read. The FB therefore does not receive any data from the partner.</li> <li>1..4294967295: Not relevant for the receiving behavior.</li> </ul>
Configuration	Input	<a href="#">"LCom_typeConfig"</a>	FB configuration
connected	Output	Bool	<p>Without LCom protocol, ("configuration.connection.comService" = "LCOM_TCP_CONNECTION");</p> <ul style="list-style-type: none"> <li>TRUE: The TCP connection to the partner is established.</li> <li>FALSE: The TCP connection to the partner is not established.</li> </ul> <p>With LCom protocol, ("configuration.connection.comService" = "LCOM_LCOM_CONNECTION");</p> <ul style="list-style-type: none"> <li>TRUE: TCP connection to partner is established and configuration data is negotiated successfully</li> <li>FALSE: TCP connection to partner is not established or waiting for negotiation of configuration data.</li> </ul>
sending	Output	Bool	<ul style="list-style-type: none"> <li>TRUE: Sends the data connected to the "sendBuffer" input parameter. Do not change send data.</li> <li>FALSE: Data that is connected at the "sendBuffer" input parameter is not sent.</li> </ul>
dataReceived	Output	Bool	<ul style="list-style-type: none"> <li>TRUE: New data was received. The value is present for one cycle.</li> <li>FALSE: No data is available for the user.</li> </ul>
receivedDataLength	Output	UDInt	<ul style="list-style-type: none"> <li>Returns the received data length in bytes.</li> </ul>
busy	Output	Bool	<ul style="list-style-type: none"> <li>TRUE: The block processing is done automatically. No intervention is required from the user.</li> <li>FALSE: There is no block processing. Intervention is required from the user. Passing parameters may be incorrect. More detailed information is provided by the output parameter "status" or the diagnostic buffer.</li> </ul>
error	Output	Bool	<ul style="list-style-type: none"> <li>TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer.</li> <li>FALSE: No error has occurred.</li> </ul>
status	Output	Word	Status of the FB, see Section <a href="#">2.5.2</a>
sendBuffer	InOut	VARIANT	Send data (array of bytes)
receiveBuffer	InOut	VARIANT	Receive data (array of bytes)
diagnostics	InOut	<a href="#">"LCom_typeDiagnostics"</a>	Diagnostics information

## 2.3 PLC Data Types

### LCom\_typeConfig

The following table shows the structure of the PLC data type "LCom\_typeConfig".

Table 2-5: PLC data type LCom\_typeConfig

Name	Type	Start value	Comment
connection	Struct		
interfacelD	HW_ANY	64	Hardware identifier of the local interface <ul style="list-style-type: none"> <li>0: if connectType = 10 (QDN)</li> </ul>
connectionID	CONN_ANY	16#0FFF	Reference to the connection 16#0001..16#0FFF
comService	USInt	2	Configuration of the communication protocol: <ul style="list-style-type: none"> <li>1: The standard TCP protocol is used.</li> <li>2 (default): The LCom protocol is used. With the LCom protocol, additional communication functionalities are enabled</li> </ul>
isClient	Bool	FALSE	<ul style="list-style-type: none"> <li>TRUE: Active connection establishment as TCP client.</li> <li>FALSE: Passive connection establishment as TCP server.</li> </ul>
connectType	USInt	1	Connection type: <ul style="list-style-type: none"> <li>1 (default) = IPv4</li> <li>10 = QDN (Qualified Domain Name)</li> </ul>
localPort	UInt	3456	Local port, see system function TCON.
partnerPort	UInt	3456	Only when the connection is active ("configuration.connection.isClient" = TRUE): Port of the partner-side, see system function TCON.
partnerIP	IP_V4	-	IP address of the partner
partnerQDN	String [254]	"	Fully qualified domain name Must finish with "."
acceptUnknownPartner	Bool	TRUE	Only for passive connection establishment ("configuration.connection.isClient" = FALSE): <ul style="list-style-type: none"> <li>TRUE: Previously not configured connection partner is accepted</li> <li>FALSE: Only the configured connection partner is accepted</li> </ul>
lifeSignCycleTime	Time	T#1s	Vital signs cycle T#1ms..T#24d20h30m20s630ms
sender	Struct		
cycleTime	Time	T#1s	Send cycle T#0ms..T#24d20h30m20s630ms
ackTimeout	Time	T#1s	Only relevant when using the LCom protocol ("configuration.connection.comService" = 2): The tag "ackTimeout" determines the monitoring time after which an acknowledgement of the sent packet is expected at the latest. If this period is exceeded, the connection is closed and re-established. T#1ms..T#24d20h30m20s630ms

## 2 LCom

Name	Type	Start value	Comment
timeSync	struct		Only relevant when using the LCom protocol ("configuration.connection.comService" = 2)
usePartnerTimestamps	Bool	FALSE	<ul style="list-style-type: none"> <li>TRUE: The received time is taken over as system time</li> <li>FALSE: Received time telegrams are ignored</li> </ul>
sendMode	USInt	0	Send mode time synchronization: <ul style="list-style-type: none"> <li>0: Inactive</li> <li>1: Cyclic time synchronization</li> <li>2: Time when the time synchronization should take place</li> </ul>
cycleTime	Time	T#1h	Only relevant if "sendMode" = 1; Send cycle time of the timestamps T#1 ms..T#24d20h30m20s630ms
sendAtTimeOfDay	Time_Of_Day	TOD#05:00:0.000	Only relevant with "sendMode" = 2; Time at which a time synchronization telegram is sent.

### LCom\_typeDiagnostics

The following table shows the structure of the PLC data type "LCom\_typeDiagnostics".

Table 2-6: PLC data type LCom\_typeDiagnostics

Name	Type	Comment
localConfig	Struct	
connection	Struct	
interfacelD	HW_ANY	Hardware identifier of the local interface
connectionID	CONN_ANY	Reference to the local connection
comService	USInt	<ul style="list-style-type: none"> <li>1: The standard TCP protocol is used</li> <li>2: The LCom protocol is used</li> </ul>
isClient	Bool	<ul style="list-style-type: none"> <li>TRUE: Active connection establishment as TCP client</li> <li>FALSE: Passive connection establishment as TCP server</li> </ul>
connectType	USInt	<ul style="list-style-type: none"> <li>1 = IPv4</li> <li>10 = QDN (Qualified Domain Name)</li> </ul>
localPort	UInt	Local port
localIP	IP_V4	Local IP address
partnerPort	UInt	Partner-side port
partnerIP	IP_V4	IP address of the partner
partnerQDN	String [254]	Fully qualified domain name
acceptUnknownPartner	Bool	<ul style="list-style-type: none"> <li>TRUE: Previously not configured connection partner is accepted</li> <li>FALSE: Only the configured connection partner is accepted</li> </ul>
useLComProtocol	Bool	<ul style="list-style-type: none"> <li>TRUE: The LCom protocol is used</li> <li>FALSE: The LCom protocol is not used</li> </ul>
lifeSignCycleTime	Time	Local vital signs cycle.

## 2 LCom

Name	Type	Comment
sender	Struct	
cycleTime	Time	Local send cycle
ackTimeout	Time	The tag <i>ackTimeout</i> determines the local monitoring time after which an acknowledgement of the sent packet is expected at the latest. If this time is exceeded, the connection is closed and re-established.
timeSync	Struct	
usePartnerTimestamps	Bool	<ul style="list-style-type: none"> <li>• TRUE: The received time is adopted as the system time.</li> <li>• FALSE: Received time telegrams are ignored.</li> </ul>
sendMode	USInt	Send mode time synchronization: <ul style="list-style-type: none"> <li>• 0: Inactive</li> <li>• 1: Cyclic time synchronization</li> <li>• 2: Time when the time synchronization should take place</li> </ul>
cycleTime	Time	Send cycle time of the timestamps
sendAtTimeOfDay	Time_Of_Day	Time at which a synchronization telegram is sent
sizeOfSendBuffer	UDInt	Size of the local send buffer in bytes
sizeOfReceiveBuffer	UDInt	Size of the local receive buffer in bytes
partnerConfig	Struct	
connection	Struct	
lifeSignCycleTime	Time	Partner vital signs cycle
sender	Struct	
cycleTime	Time	Partner send cycle
ackTimeout	Time	The tag <i>ackTimeout</i> determines the partner monitoring time after which an acknowledgement of the sent packet is expected at the latest. If this time is exceeded, the connection is closed by the partner and re-established.
timeSync	Struct	
usePartnerTimestamps	Bool	<ul style="list-style-type: none"> <li>• TRUE: The received time is taken over as system time</li> <li>• FALSE: Received time telegrams are ignored</li> </ul>
sendMode	USInt	Send mode time synchronization: <ul style="list-style-type: none"> <li>• 0: Inactive</li> <li>• 1: cyclic time synchronization.</li> <li>• 2: Time when the time synchronization should take place</li> </ul>
sizeOfSendBuffer	UDInt	Size of the send buffer in bytes of the partner
sizeOfReceiveBuffer	UDInt	Size of the receive buffer in bytes of the partner



## 2 LCom

Name	Type	Comment
statistics	Struct	Statistics
avgCallCycle	Real	Mean value between two FB calls [ms]
avgReceiveMsgCycle	Real	Mean value in which cycle data was received [ms]
maxReceiveMsgCycle	Time	Maximum occurred receive cycles
LComProtocolUsed	Bool	<ul style="list-style-type: none"> <li>TRUE: The LCom protocol is used. With the LCom protocol additional communication functionalities are enabled</li> <li>FALSE: The TCP transport protocol is used</li> </ul>
activeLComVersion	USInt	Active LCom protocol version
avgMsgSendingTime	Real	Average value of how long a sending process ( <i>sending</i> = TRUE) takes [ms]
maxMsgSendingTime	Time	Maximum value of a sending process ( <i>sending</i> = TRUE) [ms]
avgMsgReceivingTime	Real	Mean value of how long a receiving process takes [ms]
maxMsgReceivingTime	Time	Maximum value of a receiving process [ms]
numberOfSentMessages	UDInt	Number of messages sent, since the L/H edge at the <i>enable</i> input
numberOfReceivedMessages	UDInt	Number of received messages, since the L/H edge at the <i>enable</i> input
totalAckTimeouts	UInt	Number of times the monitoring time was exceeded ( <i>diagnostics.localConfig.sender.ackTimeout</i> )
totalSendCycleViolations	UInt	Number of times the send cycle is exceeded ( <i>diagnostics.localConfig.sender.cycleTime</i> )
totalReceiveCycleViolations	UInt	Number of times the receive cycle is exceeded ( <i>diagnostics.partnerConfig.sender.cycleTime</i> )
totalReconnects	UInt	Number of times the FB has reestablished the TCP connection
lastConnect	DTL	Time when the connection was successfully established to the partner
lastTimeSync	DTL	Time of the last time synchronization
bufferIndex	USInt	Index to the last diagnostic entry
buffer	Array[0..63] of Struct	Diagnostic buffer
status	Word	Status message from FB
timestamp	DTL	Timestamp at appearance of the message
isActive	Bool	<ul style="list-style-type: none"> <li>TRUE: State is still available</li> <li>FALSE: State is no longer available</li> </ul>
subFunctionErrorID	Word	If necessary, Return value of a system function
additionalValue1	Real	Additional value 1
additionalValue2	Real	Additional value 2
additionalValue3	Real	Additional value 3
additionalValue4	Time	Additional value 4

## 2.4 Integration into the User Project

You will find an application example for integrating the LCom blocks into your user application in the master copies of this library.

Additionally, you will find a detailed application example for the integration of the library into your user project in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/48955385>

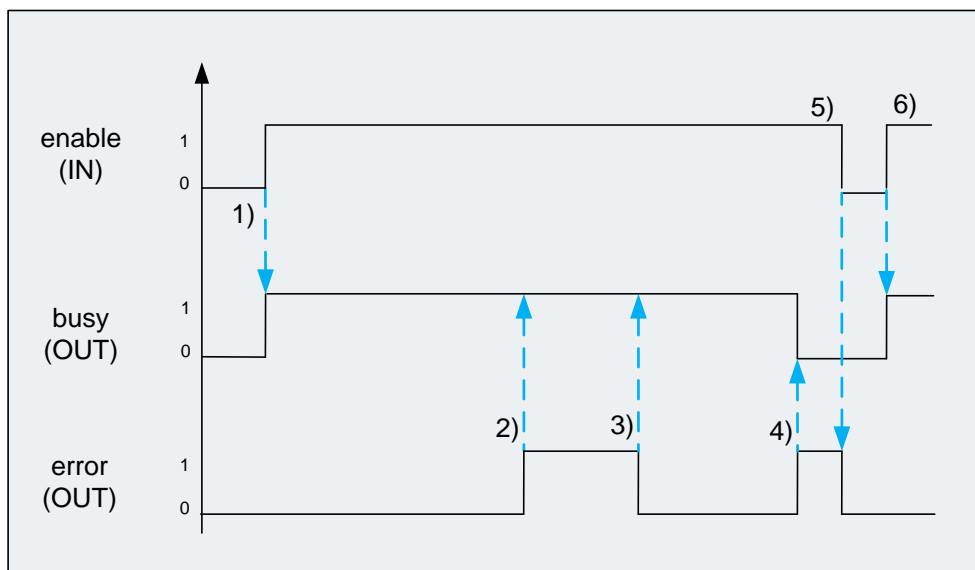
## 2.5 Error Handling

### 2.5.1 Status Outputs

An error is indicated by setting the boolean tag "error". A distinction must be made between whether the cause of the error can be remedied by the user or by the block. If it is an error that can be corrected by the block, "busy" remains set. If an error occurs that must be corrected by the user, "busy" is reset (block processing deactivated).

The following figure shows the two scenarios as a signal flow diagram.

Figure 2-5 Signal flow diagram error/busy



1. "busy" is set with a rising edge of "enable".
2. An error occurs and "error" is set. Since this is an error that can be corrected by the block itself, "busy" remains set.
3. After correcting the cause of the error (e.g. re-establishing the connection), "error" is reset.
4. An error that can only be corrected by the user occurs. Here "error" is set and "busy" is reset.
5. The pending error, which must be remedied by the user, can only be cleared with a falling edge at "enable".
6. The block is started again with a rising edge of "enable".

## Output parameters diagnostics

In the output parameter "diagnostics", there are different substructures that are defined by the PLC data type "LCom\_typeDiagnostics". The diagnostic buffer is discussed below. The complete description of the PLC data type "LCom\_typeDiagnostics" can be found in Section [2.3](#).

Various status and error messages are entered in the diagnostic buffer. The array is set to 64 entries. This array operates as a ring buffer. The tag "bufferIndex" points to the last (current) entry.

Each entry consists of the following elements:

- Status or error number
- Date and time of occurrence
- Current state of the error

The state when entering the buffer is always active. When resetting the error, the status in the buffer is set to inactive. The structure contains the return value of the system function as well as four additional values that can contain detailed information depending on the error that occurred.

Table 2-7 Structure of the diagnostic buffer

Name	Data type	Description
bufferIndex	USInt	Index to last entry
buffer	Array [0..63] of Struct	Diagnostic buffer
status	Word	Status from FB
timestamp	DTL	Timestamp at appearance of the message
isActive	Bool	TRUE: State is still present. FALSE: State is no longer present.
subFunctionErrorID	Word	If necessary, Return value of a system function
additionalValue1	Real	Additional value 1
additionalValue2	Real	Additional value 2
additionalValue3	Real	Additional value 3
additionalValue4	Time	Additional value 4

## 2.5.2 Status Messages

Table 2-8: Status messages

Code	Meaning
16#7000	No job in process.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7001	First call after receipt of a new job (rising edge at <i>enable</i> ).
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7002	Follow-up call during active FB processing.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7003	FB tries to connect as TCP client.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7004	FB is configured as TCP server and waits for a TCP client request.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7005	The TCP connection to the partner is established.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms

Code	Meaning
16#7006	The FB has successfully exchanged configuration data with the partner via LCom protocol V1. The maximum possible data length is 64 KByte.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7007	The FB has successfully established the connection and exchanged the configuration data (LCom protocol V2) with the partner. The maximum possible data length is 16 MByte.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7008	The FB terminates the connection.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7008	The FB terminates the connection.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7600	The specified send length ( <i>sendDataLength</i> ) is too large and is automatically limited by the FB. The maximum possible send length is determined by the size of the own send buffer ( <i>sendBuffer</i> ).
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : <i>sendDataLength</i> <i>additionalValue2</i> : <i>sendBuffer</i> <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms

Code	Meaning
16#7610	The specified time ( <i>configuration.connection.lifeSignCycleTime</i> ) for connection monitoring is too large and is automatically limited to 65535 ms.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Limitation to 65535 <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : Value of the user
16#7611	The specified time for the send cycle ( <i>configuration.sender.cycleTime</i> ) is too large and is automatically limited to 65535 ms.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Limitation to 65535 <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : Value of the user
16#7612	The configured monitoring time ( <i>configuration.sender.ackTimeout</i> ) is too large and is automatically limited to 65535 ms.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Limitation to 65535 <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : Value of the user
16#7613	The configured send cycle ( <i>configuration.sender.CycleTime</i> ) cannot be kept because the previous sending process has not yet been completed.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : Previous time of the previous send job

Code	Meaning
16#7614	When using the LCom protocol ( <i>configuration.connection.comService = 2</i> ): The specified send length ( <i>sendDataLength</i> ) is larger than the receive buffer of the partner.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : <i>sendDataLength</i> <i>additionalValue2</i> : Receive buffer of the partner-side <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms
	Without using the LCom protocol ( <i>configuration.connection.comService = 1</i> ): The specified <i>readDataLength</i> at the FB is larger than the specified receive buffer (receiveBuffer). The <i>readDataLength</i> is limited to the size of the specified receive buffer.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : User-defined receive buffer <i>additionalValue2</i> : <i>readDataLength</i> <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#7615	No time synchronization is performed.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..4</i> : 0.0
16#8200	The specified communication service ( <i>configuration.connection.comService</i> ) is invalid → see system function TCON.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Value of the user <i>additionalValue2...3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8201	The specified connection reference ( <i>configuration.connection.connectionID</i> ) is invalid → see system function TCON.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : T#0ms

Code	Meaning
16#8202	The specified hardware identifier for the local interface ( <i>configuration.connection.interfaceID</i> ) is invalid → see system function TCON.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8203	The specified local port ( <i>configuration.connection.localPort</i> ) is not allowed → see system function TCON.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : TCON status <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8204	The specified partner IP address ( <i>configuration.connection.partnerIP</i> ) is invalid → see system function TCON.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : TCON status <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8205	The specified mode for time synchronization ( <i>configuration.timeSync.sendMode</i> ) is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8206	The specified cycle time for time synchronization ( <i>configuration.timeSync.cycleTime</i> ) is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : Value of the user



Code	Meaning
16#8207	The specified time for the time synchronization ( <i>configuration.timeSync.sendAtTimeOfDay</i> ) is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8208	The value passed to the input tag <i>sendBuffer</i> is not of type ARRAY of BYTE and is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8209	The value passed to the output tag <i>receiveBuffer</i> is not of type ARRAY of BYTE and is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8210	The specified time ( <i>configuration.connection.lifeSignCycleTime</i> ) for connection monitoring is invalid:
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : Value of the user
16#8211	The specified send cycle ( <i>configuration.sender.cycleTime</i> ) is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : Value of the user
16#8212	The configured monitoring time ( <i>configuration.sender.ackTimeout</i> ) is invalid.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : Value of the user

Code	Meaning
16#8213	The configured connection type ( <i>configuration.connection.connectType</i> ) is invalid
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Value of the user <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8600	An error has occurred when using the TCON system function, see <i>subFunctionErrorID</i> in the diagnostic buffer and system documentation. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : TCON status <i>additionalValue1</i> : 0.0 <i>additionalValue2</i> : 0.0 <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8601	An error has occurred when using the TCON system function, see <i>subFunctionErrorID</i> in the diagnostic buffer and system documentation. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : TCON status <i>additionalValue1</i> : Local port <i>additionalValue2</i> : 0.0 <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8602	An error occurred while using the system function TSEND, see <i>subFunctionErrorID</i> in the diagnostic buffer and system documentation. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : TSEND status <i>additionalValue1</i> : 0.0 <i>additionalValue2</i> : Length of send data <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8603	An error has occurred when using the TRCV system function, see <i>subFunctionErrorID</i> in the diagnostic buffer and system documentation. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : Status of TRCV <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms

Code	Meaning
16#8604	An error has occurred when using the TDISCON system function, see <i>subFunctionErrorID</i> in the diagnostic buffer and system documentation. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : Status of TDISCON <i>additionalValue1..3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8610	The vital signs of the partner were not received in time. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Vital signs cycle <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : Time in which the FB has not received any data from the partner.
16#8611	The monitoring time of a sent data packet has expired without an acknowledgement being received from the partner. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Monitoring time <i>additionalValue2..3</i> : 0.0 <i>additionalValue4</i> : Time in which the FB has not received any data from the partner.
16#8612	An invalid acknowledge number was received from the partner, the connection will be re-established.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : Received acknowledge message number <i>additionalValue2</i> : Send acknowledge message number <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms
16#8613	The LCom (protocol) versions of the library LCom do not match the partner or an invalid (protocol) version was received. The connection is reestablished.
	Diagnostic entry: <i>timestamp</i> : Time of the event <i>isActive</i> : State <i>subFunctionErrorID</i> : 16#0 <i>additionalValue1</i> : received version <i>additionalValue2</i> : local, active version <i>additionalValue3</i> : 0.0 <i>additionalValue4</i> : T#0ms

Code	Meaning
16#8614	A telegram was received which has an ID that was not expected. The connection is reestablished.
	Diagnostic entry: <i>timestamp:</i> Time of the event <i>isActive:</i> State <i>subFunctionErrorID:</i> 16#0 <i>additionalValue1:</i> received ID <i>additionalValue2..3:</i> 0.0 <i>additionalValue4:</i> T#0ms

If errors occur during parameterization (16#8200 to 16#83FF), intervention by the user is required. The user must replace the faulty value with a permissible value and restart the function block with a rising edge at the "enable" input.

In case of an internal error (16#8600 to 16#87FF) the function block will automatically close the connection and try to re-establish it. A new rising edge at the "enable" input is not necessary.

# 3 LFTP

## 3.1 Overview

### 3.1.1 Range of Functions

A simple protocol that works according to the client/server principle and which meets the demands of this task is the **File Transfer Protocol (FTP)**.

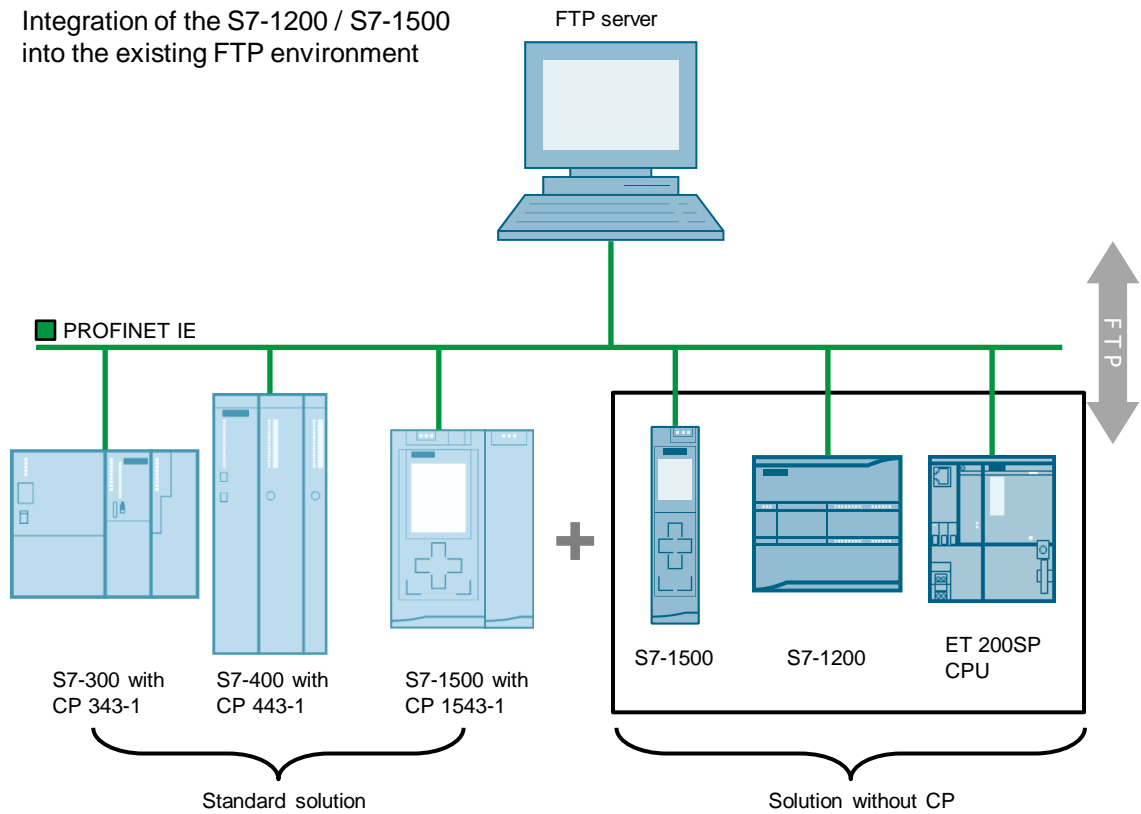
FTP allows you to store data on server systems. FTP supports almost all server systems and operating systems.

The controllers from the SIMATIC S7-300, S7-400 and S7-1500 product families support FTP communication with the help of specialized communications processors (CPs).

With this library you can implement FTP communication with an S7-1200, S7-1500 or ET200SP CPU based on Open User Communication **without a special CP**.

Figure 3-1

Integration of the S7-1200 / S7-1500 into the existing FTP environment



### 3.1.2 Components of the Library

#### Function blocks

Table 3-1: Function blocks

Name	Version	Description
LFTP_Client	V6.1.1	Organizes connection setup of the FTP control connection and data connection as well as sending and receiving of files.

#### PLC data types

Table 3-2: PLC data types

Name	Version	Description
LFTP_typeConnParam	V1.0.0	This data type describes all parameters of the connection to the FTP server.
LFTP_typeFtpParam	V1.0.0	This data type describes all parameters of the FTP commands.

### 3.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 controllers with firmware V2.9 or higher
- SIMATIC S7-1200 controllers with firmware V4.5 or higher
- SIMATIC ET 200SP Open Controllers with firmware V2.9 or higher
- CM 1542-1

## 3.2 LFTP\_Client

### 3.2.1 Interface Description

#### Description

The FTP function block "LFTP\_Client" implements FTP on the basis of Open User Communication. It can execute the following FTP commands:

- CONNECT (connect and log on)
- DISCONNECT (disconnect and log off)
- STORE (save data)
- APPEND (append data)
- RETRIEVE (retrieve data)
- DELETE (delete a file)

The FB supports both modes Active FTP and Passive FTP.

#### Parameters

Figure 3-2: LFTP\_Client

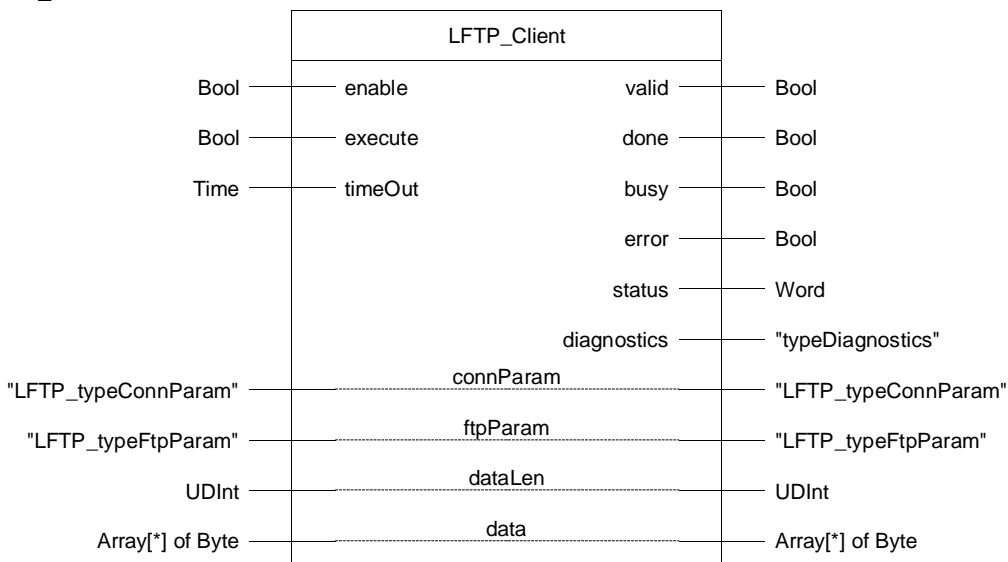


Table 3-3: Parameters of LFTP\_Client

Name	Declaration	Data type	Comment
enable	IN	Bool	TRUE = control connection to the FTP server is connected FALSE = control connection is terminated
execute	IN	Bool	FTP job trigger on rising edge
timeOut	IN	Time	Time after which a job should be automatically canceled
valid	OUT	Bool	TRUE: The FB executes its function without error.
done	OUT	Bool	TRUE: The current job (send message or activate shell) was completed successfully.
busy	OUT	Bool	TRUE: The FB is busy.

Name	Declaration	Data type	Comment
error	OUT	Bool	TRUE: An error has occurred. If "busy" is TRUE at the same time, the block attempts to correct the error itself.
status	OUT	Word	Status and error codes (see Section <a href="#">6.2.3</a> )
diagnostics	OUT	<a href="#">typeDiagnostics</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
connParam	IN_OUT	<a href="#">LFTP_typeConnParam</a>	Sets connection parameters. The hardware ID, URL of the FTP server, username and password are defined here.
ftpParam	IN_OUT	<a href="#">LFTP_typeFtpParam</a>	FTP parameters
dataLen	IN_OUT	UDInt	Length of data to be sent or received
data	IN_OUT	Array[*] of Byte	Data to be sent or received

### 3.2.2 Error Handling

Table 3-4: Status and error codes

Status	Meaning
16#0000	FTP command successfully executed
16#7000	Idling, no active connections
16#7001	First call of the block
16#7002	Control connection is being established
16#7003	Control connection is being terminated
16#7004	Control connection is established, no job active
16#7005	FTP command is being executed
16#8202	Invalid URL
16#8400	Error received from FTP server. Some possible causes could be that the login credentials are wrong or that the file name does not exist. The FTP-specific error code is output at "diagnostics.subfunctionStatus". The meaning of the FTP-specific error code can be found in the FTP specification.
16#8600	The FB is in an invalid state.
16#8601	Error in subordinate instruction "TCON" The error code of the instruction is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8602	Error in subordinate instruction "TSEND" The error code of the instruction is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8603	Error in subordinate instruction "TRCV" The error code of the instruction is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8604	Error in subordinate instruction "TCONSettings" The error code of the instruction is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8605	Error in subordinate instruction "MOVE_BLK_VARIANT" The error code of the instruction is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8086	Error in subordinate instruction "RDREC" The error code of the instruction is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8D11	Control connection not connected; data connection cannot be opened



Status	Meaning
16#8D26	Data connection cannot be opened
16#8D28	FTP server timeout
16#8DF1	Request timeout
16#8DF2	Unknown reply code
16#8DF3	Unknown command
16#8DF4	Port number below 2000
16#8F62	Action aborted; data connection will be terminated
16#8F69	Connection attempt on existing connection

**NOTE**

If the FB outputs "done" but no data after a "RETRIEVE" command, this may be caused by the communication load being set too low. Increase the communication load in the device configuration.

### 3.3 PLC Data Types

#### LFTP\_typeConnParam

This data type describes all the parameters required for the connection to the FTP server.

Table 3-5: Parameters of LFTP\_typeConnParam

Name	Type	Comment
hwID	HW_ANY	Hardware ID of the IE interface module
serverAddress	String	IP address or hostname of the FTP server
username	String	FTP user name
password	String	FTP user password

#### LFTP\_typeFtpParam

This data type describes all FTP parameters necessary for the data connection.

Table 3-6: Parameters of LFTP\_typeFtpParam

Name	Type	Comment
ftpActiveMode	BOOL	FTP mode; TRUE: active, FALSE: passive
portActiveMode	UInt	Local port for the FTP active mode, must be set to a value higher than 2000.
ftpCmd	Int	FTP command <ul style="list-style-type: none"> <li>• 2: STORE</li> <li>• 3: RETRIEVE</li> <li>• 4: DELETE</li> <li>• 6: APPEND</li> </ul>
filename	String	File name

## 3.4 Integration into the user project

You will find a detailed application example for the integration of the library into your user project in Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/81367009>

## 4 LHTTP

### 4.1 Overview

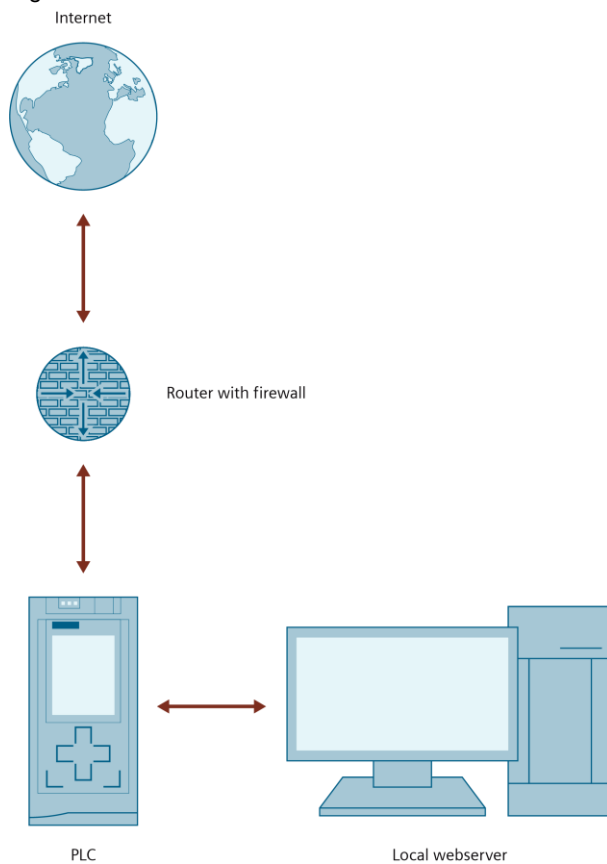
#### 4.1.1 Range of Functions

The Hypertext Transfer Protocol (HTTP) is a data transfer protocol used primarily to load Web pages from the World Wide Web.

Due to the increasing networking of plants and the advancement of the Internet of Things (IoT), HTTP and HTTPS also play an increasingly important role in automation technology.

The library for HTTP communication (LHTTP) enables the data exchange of a SIMATIC S7-1200/1500 CPU via the integrated Ethernet interface with another device in the local network or a web server in the internet via HTTP respectively HTTPS.

Figure 4-1: Overview



The LHTTP library provides function blocks with which the most conventional HTTP request methods can be implemented in the user program:

- GET
- POST
- PUT

Due to the integrated certificate management in the TIA Portal, it is also possible to transfer data securely with HTTPS using the same blocks.

## 4.1.2 Components of the Library

### Blocks

Table 4-1: Blocks of the library

Name	Type	Version	Description
LHTTP_Get	FB	V2.1.1	Realizes the HTTP method GET
LHTTP_PostPut	FB	V2.1.1	Realizes the HTTP methods POST and PUT
LHTTP_FindStringInArray	FC	V1.0.0	Searches an array of chars for a given string
LHTTP_ExtractStringFromArray	FC	V1.0.0	Extracts a string between two specified text parts from an array of chars.
LHTTP_ExtractStringFromArrayExt	FC	V1.0.0	Same function as "LHTTP_ExtractStringFromArray" with advanced options

### PLC Data Types

Table 4-2: PLC library data types

Name	Version	Description
LHTTP_typeTLS	V1.0.0	Data type for transferring certificates for secure communication via HTTPS

### 4.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.4
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- CP 1543-1 as of firmware V2.0
- CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V3.2

## 4.2 LHTTP\_Get

### 4.2.1 Interface Description

#### Description

The block implements the HTTP method GET to fetch data from a web server.

#### Parameter

Figure 4-2: LHTTP\_Get

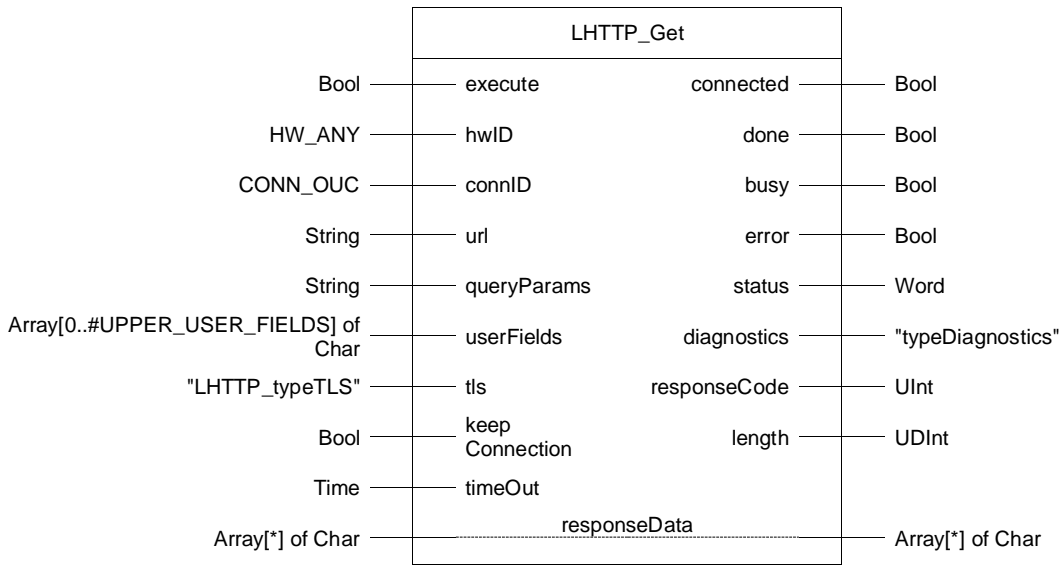


Table 4-3: Parameters of LHTTP\_Get

Name	Declaration	Data type	Description
execute	Input	Bool	With a positive edge, a connection to the web server is established and the HTTP request is sent.
hwID	Input	HW_ANY	Hardware identifier of the PN/IE interface for establishing the connection If "0" is selected, a suitable one is selected automatically.
connID	Input	CONN_OUC	Unique connection ID
url	Input	String	URL, e.g. "http://httpbin.org/get"
queryParams	Input	String	Additional query parameters if parameter "url" is not long enough, i.e. "lang=en&q=simatic".
userFields	Input	Array[0..#UPPER_USER_FIELDS] of Char	Additional user-defined header fields The header fields must be formatted according to HTTP and terminated with the "\$00" character.
tls	Input	<a href="#">"LHTTP_typeTLS"</a>	TLS certificates for secure data transmission (HTTPS) For unsafe data transmission (HTTP) leave unconnected.

Name	Declaration	Data type	Description
keepConnection	Input	Bool	TRUE: Keeps the connection established after the job has been executed
timeOut	Input	Time	Time after which a job should be automatically canceled
connected	Output	Bool	TRUE: Connection to the web server is established
done	Output	Bool	TRUE: Job finished
busy	Output	Bool	TRUE: Job is being executed
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section 4.8)
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section 12.2)
responseCode	Output	UInt	Received HTTP status code (see Table 4-10)
length	Output	UDInt	Length of the received user data
responseData	InOut	Array[*] of Char	Received user data. The array must start at "0".

### 4.2.2 Operation

The user specifies the requested resource in the form of a URL, e.g. "http://httpbin.org/get" or "https://192.168.0.1:80/index.html", at the "url" parameter.

Optional parameters can be passed with two variants:

- Appended to the URL at the parameter "url" with a "?" in front, e.g. "http://httpbin.org/get?lang=en&q=simatic"
- At the separate parameter "queryParams", e.g. "lang=en&q=simatic"; "?" or "&" is inserted automatically in between "url" and "queryParams".

With a rising edge at the "execute" parameter, the block requests the IP address belonging to the host at the configured DNS Server if necessary and establishes a connection to the web server.

If the specified URL starts with "https", a secure connection is established. In this case, the certificate of the requested web server must be referenced at the "tls" parameter (see Section 4.7). If the web server requires client authentication, the PLC's certificate must also be referenced.

The block creates the HTTP request from the specified URL and sends it to the web server.

The user data of the web server's response is output at the "responseData" parameter. The received HTTP status code is output at the "responseCode" parameter.

If the web server responds with an error (HTTP status code 4xx), no data is output.

#### NOTE

The FB does not clear the receive area at "responseData" between two request. The receive area might contain old data. Use the length of the received user data of the current request at the output "length" to only evaluate this data.

Depending on the parameter "keepConnection", the block terminates the connection to the web server after all telegrams have been successfully received or keeps it established for further requests.

**Note** The block outputs the HTTP status code received from the server with the first received telegram and writes the user data of each received telegram directly into the memory area connected to the parameter "responseData". Evaluate the user data only after the output "done" has been set.

**Note** The block does not support redirections. If the server responds with a redirection (HTTP status code 3xx), the block issues an error (see Section [4.8](#)).  
Check the URL and correct it if necessary.

### User-defined header fields

The "userFields" input allows the user to insert additional header fields into the request. The user has 256 characters available in the form of an Array of Char.

The data must be formatted according to HTTP: Two header fields are separated by an end-of-line ("R\$L"). The end of the user-defined header fields is signaled to the FB with the character "\$00". A closing line end is not necessary.

For example, the "Accept Language" and "Authorization" header fields can be inserted as follows:

```
Accept-Language: enR$LAuthorization: Basic QWxhZGRpbjpvGVuIHNlc2FtZQ==00
```

If no user-defined header fields are required, the input is not connected.

**Note** If 256 characters are not enough, the array can be increased by adjusting the local constant "UPPER\_USER\_FIELDS".

### Keep connection established

If the parameter "keepConnection" is set, the connection is not broken after the completion of a job and can be used for further requests. Especially if there are many requests to the same web server over HTTPS, this shortens the required duration of subsequent requests.

The connection is then broken by resetting "keepConnection".

## 4.3 LHTTP\_PostPut

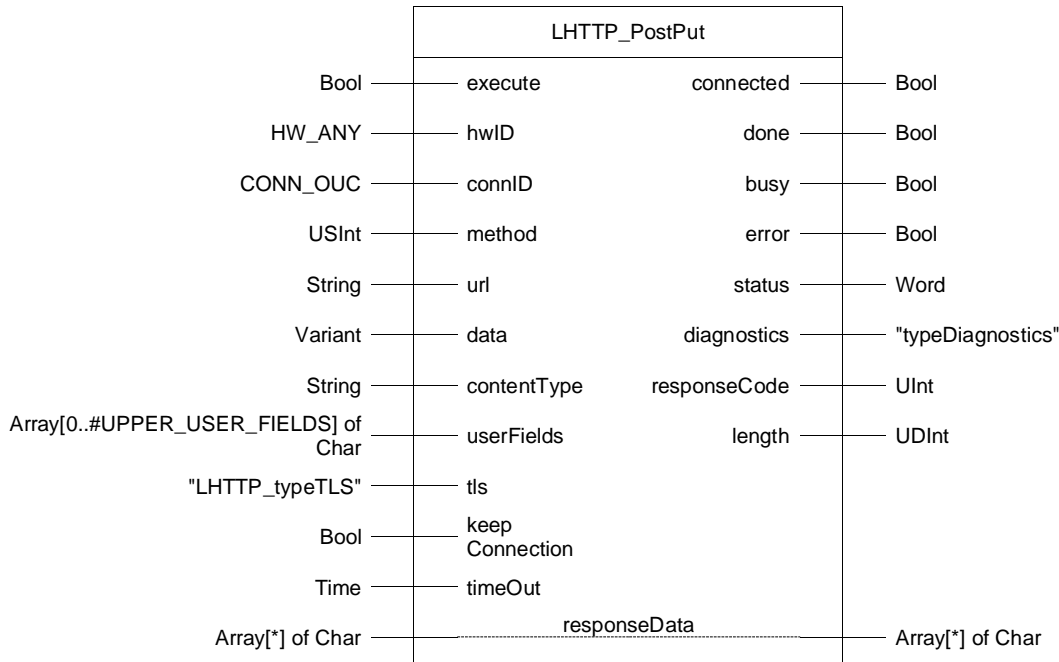
### 4.3.1 Interface Description

#### Description

The block implements the HTTP methods POST and PUT to transfer data to a web server.

#### Parameter

Figure 4-3: LHTTP\_PostPut



© Siemens AG 2023. All rights reserved.

Table 4-4: Parameters of LHTTP\_PostPut

Name	Declaration	Data type	Description
execute	Input	Bool	With a positive edge, a connection to the web server is established and the HTTP request is sent.
hwID	Input	HW_ANY	Hardware identifier of the PN/IE interface for establishing the connection If "0" is selected, a suitable one is selected automatically.
connID	Input	CONN_OUC	Unique connection ID
method	Input	USInt	HTTP method 0: POST 1: PUT
url	Input	String	URL, e.g. "http://httpbin.org/post"
contentType	Input	String	Content type of the body
data	Input	Variant	Message body Valid data types: String and Array of Char In the case of Array of Char, the data must be terminated with the character "\$00".



Name	Declaration	Data type	Description
userFields	Input	Array[0..#UPPER_USER_FIELDS] of Char	Additional user-defined header fields The header fields must be formatted according to HTTP and terminated with the "\$00" character.
tls	Input	<a href="#">"LHTTP_typeTLS"</a>	TLS certificates for secure data transmission (HTTPS) For unsafe data transmission (HTTP) leave unconnected.
keepConnection	Input	Bool	TRUE: Keeps the connection established after the job has been executed
timeOut	Input	Time	Time after which a job should be automatically canceled
connected	Output	Bool	TRUE: Connection to the web server is established
done	Output	Bool	TRUE: Job finished
busy	Output	Bool	TRUE: Job is being executed
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section <a href="#">4.8</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
responseCode	Output	UInt	Received HTTP status code (see Section <a href="#">4.8</a> )
length	Output	UDInt	Length of the received user data
responseData	InOut	Array[*] of Char	Received user data. The array must start at "0".

### 4.3.2 Operation

The user specifies the requested resource in the form of a URL, e.g. "http://httpbin.org/post" or "https://192.168.0.1:80/index.html", at the "url" parameter.

The user data can be passed as a string or, if 254 characters are not sufficient, as an array of chars at the "data" parameter. If Array of Char is used, the user data must be terminated with the character "\$00".

The content type of the body depends on the webserver and can be set with the parameter "contentType". The content type defines the format of the body.

Via the "method" parameter the user chooses whether the HTTP method POST or PUT should be used.

With a rising edge at the "execute" parameter, the block requests the IP address belonging to the host at the configured DNS Server if necessary and establishes a connection to the web server.

If the specified URL starts with "https", a secure connection is established. In this case, the certificate of the requested web server must be referenced at the "tls" parameter (see Section [4.7](#)). If the web server requires client authentication, the PLC's certificate must also be referenced.

The block creates the HTTP request from the specified URL and sends it to the server.

The user data of the server's response is output at the "responseData" parameter. The received HTTP status code is output at the "responseCode" parameter.

If the web server responds with an error (HTTP status code 4xx), no data is output.

**NOTE**

The FB does not clear the receive area at "responseData" between two request. The receive area might contain old data. Use the length of the received user data of the current request at the output "length" to only evaluate this data.

Depending on the parameter "keepConnection", the block terminates the connection to the server after all telegrams have been successfully received or keeps it established for further requests.

**Note**

The block outputs the HTTP status code received from the server with the first received telegram and writes the user data of each received telegram directly into the memory area connected to the parameter "responseData". Evaluate the user data only after the output "done" has been set.

**Note**

The block does not support redirections. If the server responds with a redirection (HTTP status code 3xx), the block issues an error (see Section [4.8](#)).

Check the URL and correct it if necessary.

**User-defined header fields**

The "userFields" input allows the user to insert additional header fields into the request. The user has 256 characters available in the form of an Array of Char.

The data must be formatted according to HTTP: Two header fields are separated by an end-of-line ("R\$L"). The end of the user-defined header fields is signaled to the FB with the character "\$00". A closing line end is not necessary.

For example, the "Accept Language" and "Authorization" header fields can be inserted as follows:

```
Accept-Language: enR$LAuthorization: Basic QWxhZGRpbjpvGVuIHNlc2FtZQ==00
```

If no user-defined header fields are required, the input is not connected.

**Note**

If 256 characters are not enough, the array can be increased by adjusting the local constant "UPPER\_USER\_FIELDS".

**Keep connection established**

If the parameter "keepConnection" is set, the connection is not broken after the completion of a job and can be used for further requests. Especially if there are many requests to the same web server over HTTPS, this shortens the required duration of subsequent requests.

The connection is then broken by resetting "keepConnection".

## 4.4 LHTTP\_FindStringInArray

### Description

The function "LHTTP\_FindStringInArray" searches an Array of Char for a string and returns its position as return value.

### Parameter

Figure 4-4: LHTTP\_FindStringInArray

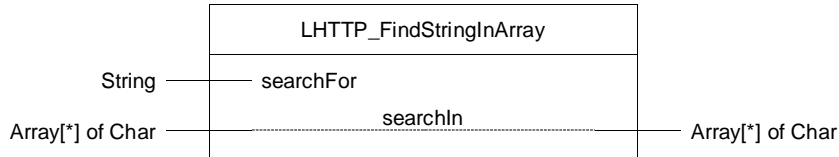


Table 4-5: Parameters of LHTTP\_FindStringInArray

Name	Declaration	Data type	Description
searchFor	Input	String	Text to search for.
searchIn	InOut	Array[*] of Char	Array of Char to be searched.
Ret_Val	Return	DInt	Position (index) of the searched string in the array. -1: String was not found.

## 4.5 LHTTP\_ExtractStringFromArray

### Description

The LHTTP\_ExtractStringFromArray function extracts a string between two specified text parts from an array of chars.

### Parameter

Figure 4-5: LHTTP\_ExtractStringFromArray

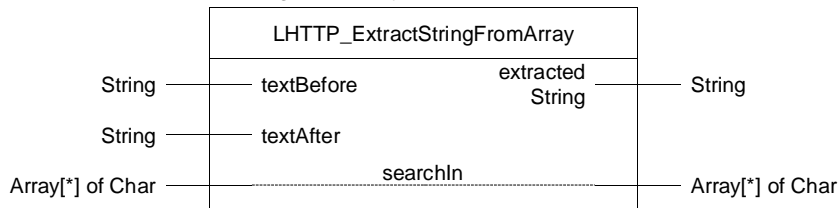


Table 4-6: Parameters of LHTTP\_ExtractStringFromArray

Name	Declaration	Data type	Description
textBefore	Input	String	Text part before the text that is to be extracted
textAfter	Input	String	Text part after the text that is to be extracted
extractedString	Output	String	Extracted Text
searchIn	InOut	Array[*] of Char	Array of Char to be searched

Name	Declaration	Data type	Description
Ret_Val	Return	Word	Return value: 16#0000: Successful, Text was found 16#9001: Only text before was found. String is output with max. length 16#9002: Neither text before nor after was found

## 4.6 LHTTP\_ExtractStringFromArrayExt

### Description

The function "LHTTP\_ExtractStringFromArrayExt" extracts a string between two specified text parts from an array of char with extended options.

**Note**

The function is part of the library to allow easy parsing of the received data, but is not used by the library blocks themselves.

### Parameter

Figure 4-6: LHTTP\_ExtractStringFromArrayExt

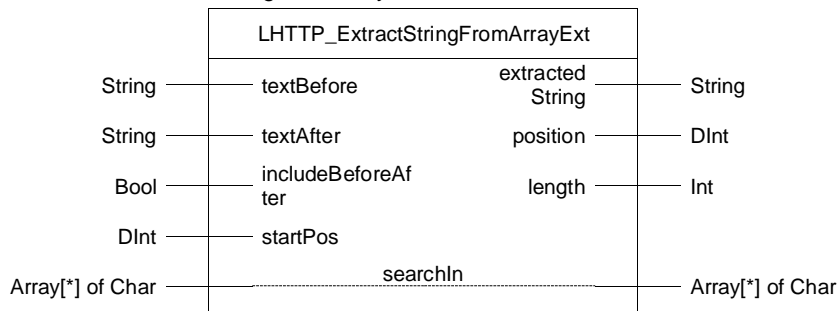


Table 4-7: Parameters of LHTTP\_ExtractStringFromArrayExt

Name	Declaration	Data type	Description
textBefore	Input	String	Text part before the text that is to be extracted
textAfter	Input	String	Text part after the text that is to be extracted
includeBeforeAfter	Input	Bool	If TRUE, the text parts before and after are extracted as well
startPos	Input	DInt	Position in the array from which the search is to be started
extractedString	Output	String	Extracted Text
position	Output	DInt	position (index) of the extracted text within the array
length	Output	Int	Length of the extracted text
searchIn	InOut	Array[*] of Char	Array of Char to be searched
Ret_Val	Return	Word	Return value: 16#0000: Successful, Text was found 16#9001: Only text before was found. String is output with max. length 16#9002: Neither text before nor after was found

## 4.7 PLC Data Types

### LHTTP\_typeTLS

The PLC data type "LHTTP\_typeTLS" references the certificates required for secure connections via HTTPS.

Table 4-8: Parameters of LHTTP\_typeTLS

Name	Data type	Description
validateServerIdentity	Bool	A set bit means that the client validates the subjectAlternateName in the server's X.509 V3 certificate to verify the server's identity. The certificates are checked when the connection is established.
serverCert	UDInt	ID of the X.509 V3 certificate (usually a CA certificate) used by the TLS client to validate the TLS server authentication. If this parameter is "0", the TLS client uses all (CA) certificates currently loaded in the client's Certificate Store to validate the server authentication.
clientCert	UDInt	ID of your own X.509 V3 certificate. This is only relevant if the TLS server requires client authentication and can otherwise remain set to "0".

**Note**

You will find an application example for the usage of certificates in TIA Portal in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/109769068>

## 4.8 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The following are the status and error codes specific to the "LHTTP" library.

The function blocks "LHTTP\_Get" and "LHTTP\_PostPut" output internal status and error codes describing the state of the function block as well as the HTTP status code received from the partner.

### Internal status codes

Table 4-9: Internal status codes

Code	Description
16#0000	Job completed without errors
16#0003	Connection broken
16#7000	No job active
16#7001	First call of the command
16#7002	Follow-up call of the command
16#7003	Connection is broken
16#7004	Connection is established and monitored. FB is waiting for job.
16#7005	Request has been sent, response pending
16#8201	The array at the parameter "responseData" does not begin with "0".
16#8202	Invalid URL. Check the spelling.

Code	Description
16#8203	The host name in the URL is different from the host to which the current connection exists. Reset "keepConnection" and run the job again.
16#8206	The end of the user-defined header fields could not be detected. Terminate the header fields with the "\$00" character.
16#8210	The internal send buffer is too small for the request to be transmitted. Adjust the size of the send buffer at the local constant "UPPER_REQUEST_ARRAY".
16#8220	The size of the array at the parameter "responseData" is not sufficient to store the received user data. The array was completely filled and the length of the received user data is output at the parameter "length".
16#8403	The web server responded with a redirection. Redirections are not supported by the LHTTP library.
16#8404	The web server responded with an error message. The received HTTP status code is output at the "responseCode" output.
16#8405	The header of the response telegram is invalid.
16#8408	Request timeout. Check the Ethernet connection of the PLC and the URL.
16#84C4	The connection to the web server was interrupted. Check the network connection.
16#84C5	Connection to the web server has broken.
16#8600	The FB is in an invalid state.
16#8601	Error in subordinate command "TSEND_C" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8603	Error of subordinate command "TRCV" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8612	Error in subordinate command "MOVE_BLK_VARIANT" when inserting user-defined header fields. The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8620	Encoding of the received message could not be detected or is not supported.
16#8630	Error of subordinate command "MOV_BLK_VARIANT" in region "TE_IDENTITY". The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8640	Chunk-coded message could not be decoded.
16#8641	Error of subordinate command MOV_BLK_VARIANT in region "TE_CHUNKED". The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.

## HTTP status codes

The following table lists common HTTP status codes:

Table 4-10: Common HTTP status codes

Status code	Message	Meaning
2xx - Successful operation		
200	OK	The request was processed successfully.
3xx - Redirection		
301	Moved Permanently	The requested resource is now available at the address specified in the "Location" header field. The old address is no longer valid.
304	Not Modified	The content of the requested resource has not changed since the last request from the client and is therefore not transferred.
4xx – Client error		
400	Bad Request	The request message was structured incorrectly.
401	Unauthorized	The request cannot be performed without valid authentication.
403	Forbidden	The request was not executed due to lack of client authorization.
404	Not Found	The requested resource is not available in the required form.
5xx – Server error		
500	Internal Server Error	Unexpected service error.
502	Bad Gateway	The server could not fulfill its functionality as a gateway or proxy because it received an invalid response.
503	Service Unavailable	The server is temporarily unavailable.
504	Gateway Timeout	The server could not perform its function as a gateway or proxy because it did not receive a response from the servers or services it was using within a specified period of time.

Further HTTP status codes can be found on Wikipedia, for example:

<https://de.wikipedia.org/wiki/HTTP-Statuscode>

## 5 LMQTT

### 5.1 Overview

#### 5.1.1 Range of Functions

The library "LMQTT" provides you with a function block to implement the MQTT protocol in SIMATIC S7-1200/1500 controllers. The FB allows you to submit MQTT messages to a broker (publisher role) and to create subscriptions (subscriber role). The communication can be secured via a TLS connection.

#### 5.1.2 Components of the Library

The "LMQTT" library contains the following objects.

#### Blocks

Table 5-1: Blocks of the library

Name	Type	Version	Description
LMQTT_Client	FB	V3.0.5	Implements the MQTT Client function and allows to submit MQTT messages to a broker (Publisher role) and create subscriptions (Subscriber role).
LMQTT_ConvertToUtf8	FC	V3.0.0	Implements a function to convert a WCHAR to UTF-8 compliant formatting.

#### PLC Data Types

Table 5-2: PLC library data types

Name	Version	Description
LMQTT_typeConnParam	V3.0.0	Contains all information to establish a connection with the MQTT Broker.

#### 5.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.4
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- CP 1543-1 as of firmware V2.0
- CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V3.2



## 5.2 LMQTT\_Client

### 5.2.1 Interface Description

#### Description

The function block "LMQTT\_Client" integrates the MQTT Client function and allows you to submit MQTT messages to a broker (Publisher role) and to create subscriptions (Subscriber role). The communication can be secured via a TLS connection.

#### Parameter

Figure 5-1: LMQTT\_Client

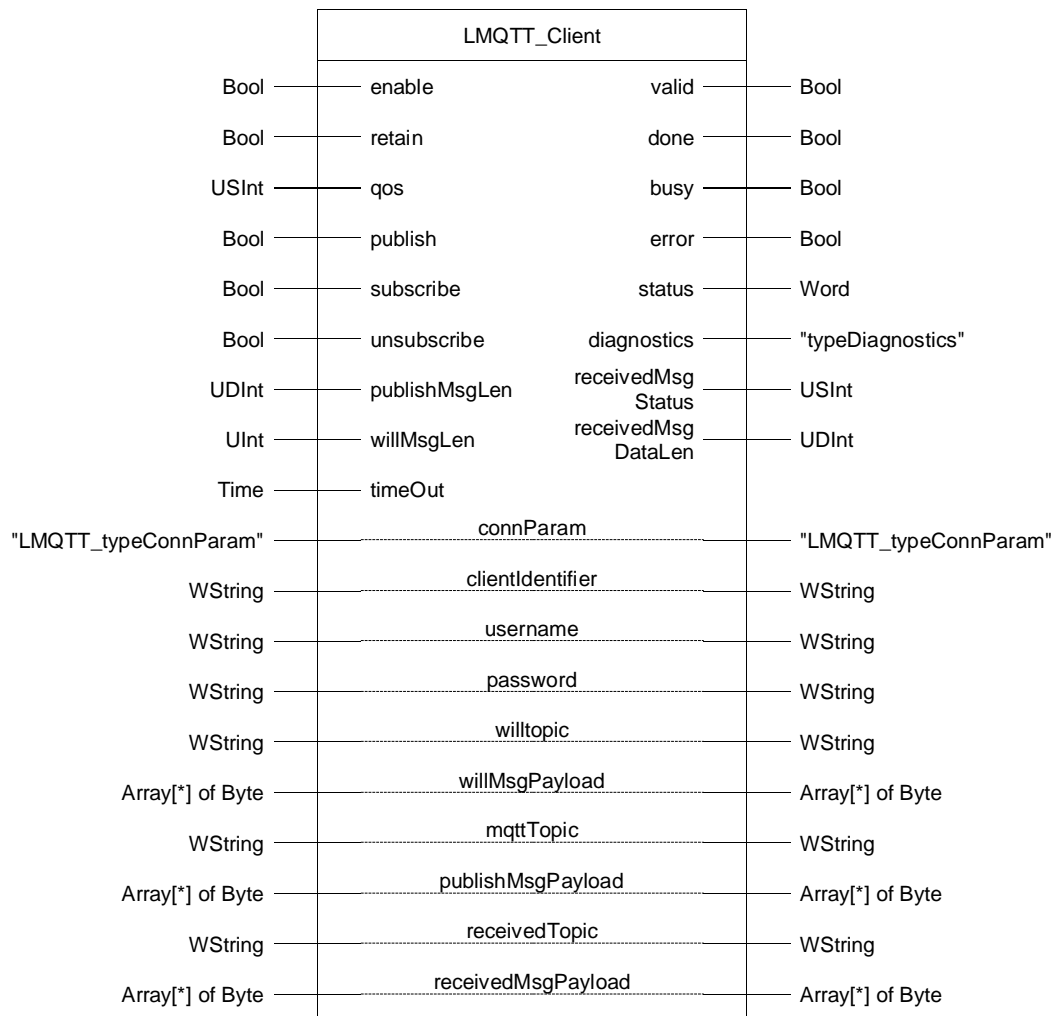


Table 5-3: Parameters of LMQTT\_Client

Name	Declaration	Data type	Comment
enable	Input	Bool	TRUE: The connection to the MQTT Broker is established and maintained. FALSE: The connection is broken.
retain	Input	Bool	TRUE: The data is sent with the "retain" flag. FALSE: The data is sent without the "retain" flag.
qos	Input	USInt	Quality of Service 0: Messages are sent or subscribed with QoS 0. 1: Messages are sent or subscribed with QoS 1. 2: Messages are sent with QoS 2 (the FB does not support QoS 2 for subscriptions).
publish	Input	Bool	With existing connection to MQTT Broker: Rising edge: Publish is executed.
subscribe	Input	Bool	With existing connection to MQTT Broker: Rising edge: Subscribe is executed.
unsubscribe	Input	Bool	With existing connection to MQTT Broker: Rising edge: Unsubscribe is executed.
publishMsgLen	Input	UDInt	Current length of valid data in the "publishMsgPayload" array parameter.
willMsgLen	Input	UInt	Current length of valid data in the array parameter "willMsgPayload".
timeOut	Input	Time	Optional parameter to configure time monitoring. After the time has expired, the current job will be considered as failed.
valid	Output	Bool	TRUE: The FB executes its function without error.
done	Output	Bool	TRUE: The current job (publish/subscribe/unsubscribe) was executed successfully.
busy	Output	Bool	TRUE: The FB is busy.
error	Output	Bool	TRUE: An error has occurred. If "busy" is TRUE at the same time, the block attempts to correct the error itself.
status	Output	Word	Status and error codes (see Section 5.5)
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section 12.2)
receivedMsgStatus	Output	USInt	Indicates for one cycle at a time when a new message has been received (subscription) 0: No new message received. 1: New valid message received. 2: New message received, but the message is invalid or received data is larger than the memory area of the receive topic or the receive message
ReceivedMsgData Len	Output	UDInt	Number of valid data in the "receivedMsgPayload" array parameter.
connParam	InOut	<a href="#">"LMQTT_typeConnParam"</a>	Parameters to establish a connection to MQTT Broker.
clientIdentifier	InOut	WString	Client identifier used when establishing a connection.
username	InOut	WString	Optional: Username for connection establishment

## 5 LMQTT

Name	Declaration	Data type	Comment
password	InOut	WString	Optional: Password for connection establishment
willtopic	InOut	WString	Optional: Topic to which the "Last Will" message is sent
willMsgPayload	InOut	Array [*] of bytes	Optional: Message sent as "Last Will".
mqttTopic	InOut	WString	MQTT topic used for publish/subscribe/unsubscribe.
publishMsgPayload	InOut	Array [*] of bytes	MQTT message that is transmitted as user data during publishing.
receivedTopic	InOut	WString	MQTT topic on which a message about the subscription was received.
receivedMsgPayload	InOut	Array [*] of bytes	User data received in the message about a subscription.

**Note**

While a job is being processed, the block does not accept any other jobs – this includes the period keep alive function. Wait until the existing job is completed by evaluating the status 16#7004 and then generate an edge at the respective input.

## 5.2.2 Error Handling

### 5.2.2.1 Block Outputs without Error

#### Send messages

Messages can be sent when the block has been successfully connected. The following table explains the values of the "status" parameter in relation to "valid", "done", and "busy" when there is no error:

Table 5-4

status	valid	busy	done	Description
16#7000	FALSE	FALSE	FALSE	Block not in process/not connected.
16#7001	TRUE	TRUE	FALSE	First call after "enable" = TRUE.
16#7002	TRUE	TRUE	FALSE	Block processes job.
16#7003	TRUE	TRUE	FALSE	MQTT is connected.
16#7004	TRUE	TRUE	FALSE	MQTT client connected and ready for job.
16#7005	TRUE	TRUE	FALSE	Ping to the broker is sent.
16#7006	TRUE	TRUE	FALSE	Publish job is executed.
16#7008	TRUE	TRUE	FALSE	Subscribe job is executed.
16#7010	TRUE	TRUE	FALSE	Unsubscribe job is executed.
16#0000	TRUE	TRUE	TRUE	Last job (publish/subscribe/unsubscribe) successfully executed.

#### Receive messages

Messages can only be received if the block is successfully connected and at least one topic has been subscribed.

The following table explains the values of the "receivedMsgStatus" parameter depending on "done" and "busy" if there is no error.

Table 5-5

receivedMsgStatus	valid	busy	status	Description
0	TRUE	TRUE	16#7004	Block connected, but no message received.
1	TRUE	TRUE	16#7004	Block connected, a new valid message has been received. receivedTopic, receivedMsgPayload, and receivedMsgDataLen can be evaluated.
2	TRUE	TRUE	16#7004	Block connected, a new invalid message was received. receivedTopic, receivedMsgPayload, and receivedMsgDataLen contain incomplete or invalid data.

## 5.2.2.2 Block Outputs in Case of Error

Via the interface parameters "error", "status", and "diagnostics" you diagnose the block. If an error is pending, "error" is set. The output "status" provides you with the associated error source of the FB. The extended diagnostic information can be found in the "diagnostics" output (see Section [12.2](#)).

**Errors that must be corrected by the user**

The error cannot be corrected automatically by the FB and the FB stops processing. After the user has corrected the error, a new edge at the "enable" input is required.

The following table explains the error sources for the status codes of the "status" output:

Table 5-6

status	error	valid	busy	Error source	subfunctionStatus
16#7000	FALSE	FALSE	FALSE	Client not connected.	-
16#8600	TRUE	FALSE	FALSE	Undefined state in the state machine.	-
16#8200	TRUE	FALSE	FALSE	Client ID is null – client identifier must be at least 1 character long.	-
16#8210	TRUE	FALSE	FALSE	Invalid array size at the "willMsgLen" parameter.	-
16#8220	TRUE	FALSE	FALSE	Invalid value at the "keepAlive" parameter.	-
16#8230	TRUE	FALSE	FALSE	Invalid value at parameter "qos".	-
16#8601	TRUE	FALSE	FALSE	Error establishing connection.	Status code of the "TCON" subordinate command
16#8620	TRUE	FALSE	FALSE	Connection establishment timeout. If necessary, increase the "timeOut" value.	-
16#8700	TRUE	FALSE	FALSE	Invalid length in received MQTT packet – new connection required.	-
16#8710	TRUE	FALSE	FALSE	Invalid QoS in received MQTT packet – new connection required.	-
16#8720	TRUE	FALSE	FALSE	Existing session at the MQTT broker. New connection required.	-
16#8730	TRUE	FALSE	FALSE	Connection rejected by MQTT broker. New connection required.	1: The MQTT Broker does not support the MQTT protocol 3.1.1 2: The MQTT broker has rejected the client identifier. 3: The MQTT service at the MQTT broker is not reachable. 4: The data in the Username or Password field is corrupted. 5: The client is not authorized to connect (username or password incorrect) 6-255: unknown error.

**Errors that are automatically corrected by the FB**

The FB attempts to correct the error automatically and outputs a status code for information.  
The following table explains the error sources for the status codes of the "status" output:

Table 5-7: Errors that are automatically corrected by the FB

status	error	valid	busy	Error source	subfunctionStatus
16#9000	TRUE	FALSE	TRUE	Connection to the MQTT broker lost. Automatic reconnection in progress.	-
16#8210	TRUE	FALSE	TRUE	Invalid array size at the "publishMsgLen" parameter.	-
16#8230	TRUE	FALSE	TRUE	Invalid value at parameter "qos".	-
16#8740	TRUE	FALSE	TRUE	MQTT packet was acknowledged by the broker with invalid packetID.	-
16#8750	TRUE	FALSE	TRUE	MQTT Subscription was rejected by the broker.	-
16#8001	TRUE	FALSE	TRUE	Simultaneous input (rising edge) at publish/subscribe/unsubscribe inputs	-
16#8300	TRUE	FALSE	TRUE	Publish Topic empty or invalid.	-
16#8310	TRUE	FALSE	TRUE	Subscribe Topic empty or invalid.	-
16#8320	TRUE	FALSE	TRUE	Unsubscribe Topic empty or invalid.	-
16#8650	TRUE	FALSE	TRUE	Timeout during the processing of the job.	-
16#8670	TRUE	FALSE	TRUE	Timeout during MQTT connection setup with the MQTT broker.	-

### 5.3 LMQTT\_ConvertToUtf8

This FC converts a WChar into a UTF-8 compliant format, which is necessary for correct transmission in MQTT Topics.

Figure 5-2: LMQTT\_ConvertToUtf8

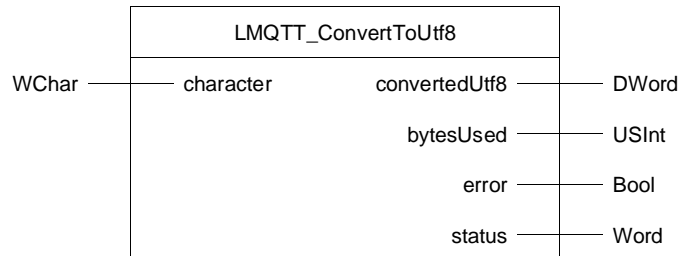


Table 5-8: Parameters of LMQTT\_ConvertToUtf8

Name	Declaration	Data type	Comment
character	Input	WChar	Character to be transformed.
convertedUtf8	Output	DWord	Single UTF-8 compliant bytes. Bytes 0-2 are used. Byte 3 is not used.
bytesUsed	Output	USInt	Number of bytes occupied by the conversion.
error	Output	Bool	Error in execution.
status	Output	Word	Status code for editing.

© Siemens AG 2023. All rights reserved.

### 5.4 PLC Data Types

#### LMQTT\_typeConnParam

This data type stores all information required to establish the MQTT connection. You can set these parameters according to your specifications.

This data type contains all information about TCP and MQTT, e.g.:

- IP address or DNS name for establishing the connection
- Logon information at the broker
- Certificates for encrypted TLS connection

The following table shows the structure of the "LMQTT\_typeConnParam" data type.

Table 5-9

Name	Data type	Meaning
hwId	HW_ANY	Hardware identifier of the PN/IE interface for establishing the connection. If "0" is selected, a suitable one is selected automatically.
connId	CONN_OUC	Connection ID for establishing the connection.

Name	Data type	Meaning
mqttBrokerAddress	Struct	Structure for the connection information of the MQTT broker.
mqttBrokerAddress.qdnAddress	String	Name (e.g. DNS name) of the MQTT broker. If this parameter is used, the IP address entry can be omitted.
mqttBrokerAddress.ipAddress	IP_V4	IP address of the MQTT Broker
mqttBrokerAddress.port	UInt	MQTT-Port
tls	Struct	Structure for the secured connection information
tls.enableTls	Bool	TRUE: Connection is secured with TLS. FALSE: Unsecured connection.
tls.validateServerIdentity	Bool	TRUE: The certificate of the MQTT Broker is validated when the connection is established. FALSE: No validation.
tls.brokerCert	UDInt	Certificate ID of the MQTT Broker certificate.
tls.clientCert	UDInt	Certificate ID of the MQTT Client certificate.
keepAlive	UInt	Keep-Alive mechanism of MQTT in seconds. 0: No Keep-Alive.

**Note**

You will find an application example for the usage of certificates in TIA Portal in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/109769068>

## 5.5 Integration into the User Project

You will find a detailed application example for the integration of the library into your user project in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/109748872>



## 6 LMindConn

### 6.1 Overview

#### 6.1.1 Range of Functions

MindSphere is the cloud-based, open IoT operating system from Siemens. Using the "MindConnect IoT Extension" upgrade, it is possible to connect devices to MindSphere over MQTT.

The "LMindConn" library implements the direct connecting of controllers through the MindConnect IoT Extension to MindSphere and uses the "LMQTT" library for the MQTT communication.

The library provides the following functions:

- Convert data points to a MindSphere-compliant message
- Request credentials from the MindConnect IoT Extension
- Register device at the MindConnect IoT Extension
- Transmit messages to the MindConnect IoT Extension (Publisher role)
- Receive messages from the MindConnect IoT Extension (Subscriber role)

The communication can be secured via a TLS connection.

**Note** The MQTT Client supports MQTT protocol version 3.1.1.

**Note** For the transmission of characters (Char) and character strings (String), all characters supported by SIMATIC Controllers (up to Unicode U-D7FF) are allowed.

#### 6.1.2 Components of the Library

The library "LMindConn" contains the following objects.

##### Blocks

Table 6-1: Blocks of the library

Name	Type	Version	Description
LMindConn_MQTT	FB	V2.0.0	Controls the communication between S7 CPU and MindSphere based on MQTT protocol.
LMindConn_ConvertDatapoint	FC	V2.0.0	Converts a data point into a MindSphere-compliant message.
LMindConn_ConvertDatapoints50	FC	V2.0.0	Converts up to 50 data points into a MindSphere-compliant message.

## PLC Data Types

Table 6-2: PLC library data types

Name	Version	Description
LMindConn_typeDatapointDefinition	V1.0.0	Describes the metadata of a data point.
LMindConn_typeUserParam	V2.0.0	Describes the parameters for connection setup and device registration.

### 6.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.4
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- CP 1543-1 as of firmware V2.0
- CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V3.2

## 6.2 LMindConn\_MQTT

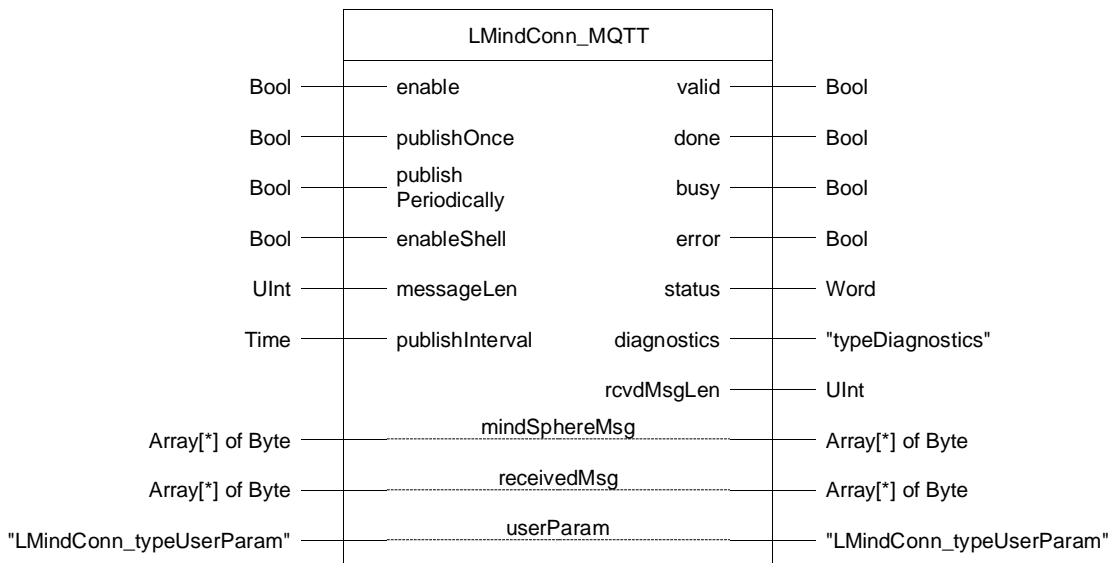
### 6.2.1 Interface Description

#### Description

The block controls the communication between the controller and MindSphere based on the MQTT protocol.

#### Parameter

Figure 6-1: LMindConn\_MQTT



## 6 LMindConn

Table 6-3: Parameter of LMindConn\_MQTT

Name	Declaration	Data type	Comment
enable	Input	Bool	TRUE: The FB establishes and maintains a connection to the MindConnect IoT Extension. The CPU is registered as a device. FALSE: The FB establishes the connection to the MindConnect IoT Extension.
publishOnce	Input	Bool	TRUE: With a positive edge, the FB sends the message at the parameter "mindSphereMsg" once to the MindSphere.
publishPeriodically	Input	Bool	TRUE: The FB sends messages periodically at the "publishInterval" interval via the "mindSphereMsg" parameter to the MindSphere.
enableShell	Input	Bool	TRUE: The FB creates a shell in the MindConnect IoT Extension and receives messages for the topic "s/ds".
messageLen	Input	UInt	Length of the "mindSphereMsg" parameter message to be sent
publishInterval	Input	Time	Time interval for periodically sending messages to MindSphere
valid	Output	Bool	TRUE: The FB executes its function without error.
done	Output	Bool	TRUE: The current job (send message or activate shell) was completed successfully.
busy	Output	Bool	TRUE: The FB is busy.
error	Output	Bool	TRUE: An error has occurred. If "busy" is TRUE at the same time, the block attempts to correct the error itself.
status	Output	Word	Status and error codes (see Section <a href="#">6.2.3</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
rcvdMsgLen	Output	UInt	Length of the received message
mindSphereMsg	InOut	Array[*] of bytes	MindSphere-compliant message to be sent
receivedMsg	InOut	Array[*] of bytes	Received message
userParam	InOut	<a href="#">"LMindConn_typeUserParam"</a>	Parameters for connection establishment and device registration. Detailed information about the structure of the PLC data type "LMindConn_typeUserParam" can be found in Section <a href="#">6.5</a> .

## 6.2.2 Operation

### Authentication

Each device must authenticate itself when connecting to the MindConnect IoT Extension.

The FB "LMindConn\_MQTT" supports two authentication options:

- Option 1: Authentication by means of previously created user data
- Option 2: Request for user data when establishing a connection, acceptance by a user in the MindConnect IoT Extension and authentication with the received user data

The FB automatically determines the type of authentication based on the parameters provided. If you fill the parameters "userParam.username" and "userParam.password", the FB authenticates itself with this user data (option 1).

If the parameters are not filled, the FB with the device ID "userParam.deviceID" requests user data from the MindConnect IoT Extension and a user can accept or reject the request (option 2).

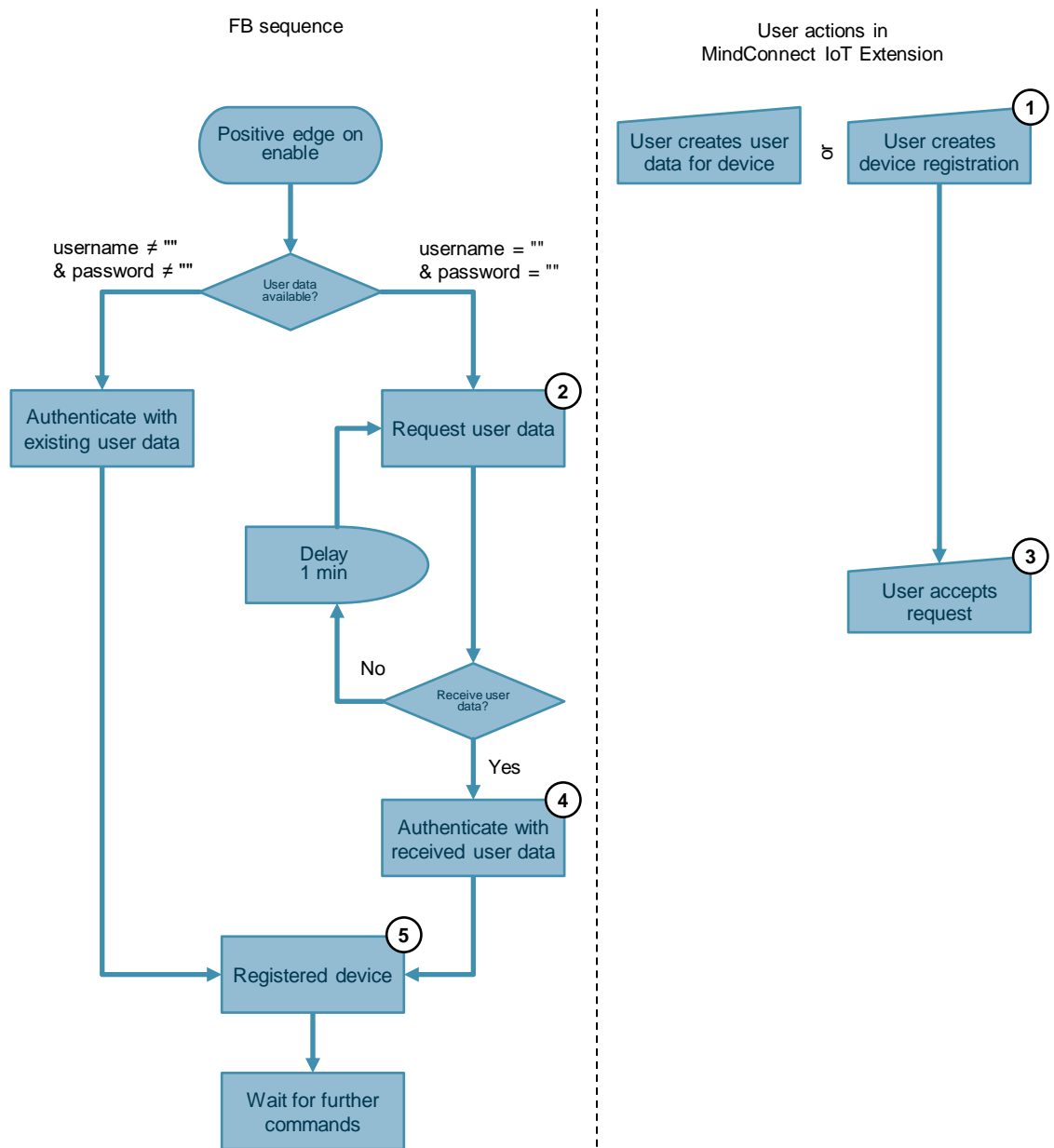
**Note**

Received user data is output via the parameters "userParam.username" and "userParam.password" and is, therefore, automatically used for authentication when a new connection is established.

To avoid having to request user data again after a reboot of the CPU, save the user data.

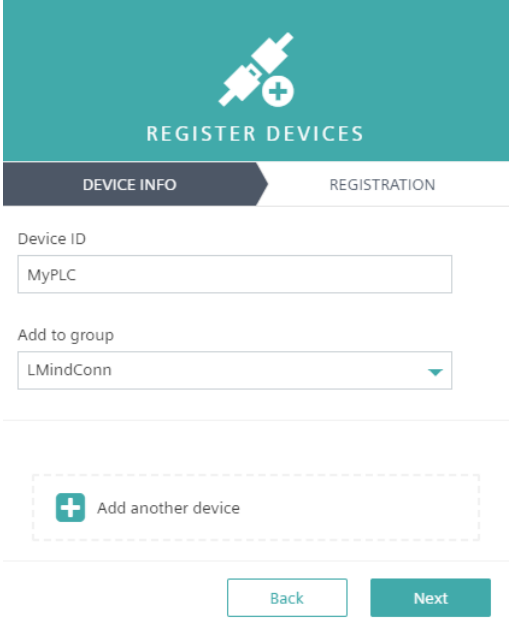
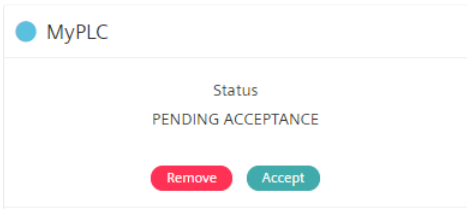
The authentication and device registration process is shown in the following flowchart:

Figure 6-2: Authentication and device registration procedure



The following table contains explanations of individual steps for the authentication option 2 from the flow chart above.

Table 6-4: Procedure request for user data and acceptance by user

	Explanation
1.	<p>A user creates a device registration with a freely selectable device ID in the MindConnect IoT Extension.</p> 
2.	<p>The FB connects to the MindConnect IoT Extension with the device ID selected by the user (parameter "userParam.deviceID") and requests user data.</p>
3.	<p>A user accepts the request for the user data of the FB in the MindConnect IoT Extension.</p> 
4.	<p>The FB disconnects from the MindConnect IoT Extension and reconnects with the received user data.</p>
5.	<p>The FB registers the CPU as a device with the device ID (parameter "userParam.deviceID") and the device type (parameter "userParam.deviceType").</p>

### 6.2.3 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The status and error codes specific to the FB "LMindConn\_MQTT" are listed below.

Table 6-5: Status and error codes

Code	Description
16#0000	The current job (send message or activate shell) was completed successfully.
16#0001	Receive new message from MindSphere
16#7000	FB is inactive
16#7001	First call after a positive edge at "enable"
16#7002	Follow-up call after a positive edge at "enable"
16#7003	The FB has requested user data from the MindConnect IoT Extension and is waiting for acceptance by a user.
16#7004	The connection was successfully established and the device registered. The FB is waiting for further commands.
16#8201	User data for authentication at the MindConnect IoT Extension is missing. Either specify the user data to "userParam.username" and "userParam.password" or specify a device ID to "userParam.deviceID" so that the FB can request user data.
16#8211	The time interval for periodically sending messages to MindSphere "publishInterval" is "0".
16#8212	The MindSphere compliant message is too long. MindConnect IoT Extension supports a maximum message length of 16,384 bytes.
16#8600	The FB is in an invalid state.
16#8601	Error from subordinate FB "LMQTT_Client". The error code of the FB is output to "diagnostics.subfunctionStatus". The meaning of the respective error code can be found in Section <a href="#">5.2.2.2</a> .
16#8602	Error from subordinate FC "LGF_DecodeUtf8" when decoding the received user data. The error code of the FC is output to "diagnostics.subfunctionStatus". The meaning of the respective error code can be found in the documentation for the Library of General Functions (LGF): <a href="https://support.industry.siemens.com/cs/ww/en/view/109479728">https://support.industry.siemens.com/cs/ww/en/view/109479728</a>
16#8603	Error of subordinate FC "LMQTT_ConvertToUtf8" when encoding the data for the device registration. The error code of the FC is output to "diagnostics.subfunctionStatus".

## 6.3 LMindConn\_ConvertDatapoint

### Description

The function converts a data point into a MindSphere-compliant message.

### Parameter

Figure 6-3: LMindConn\_ConvertDatapoint

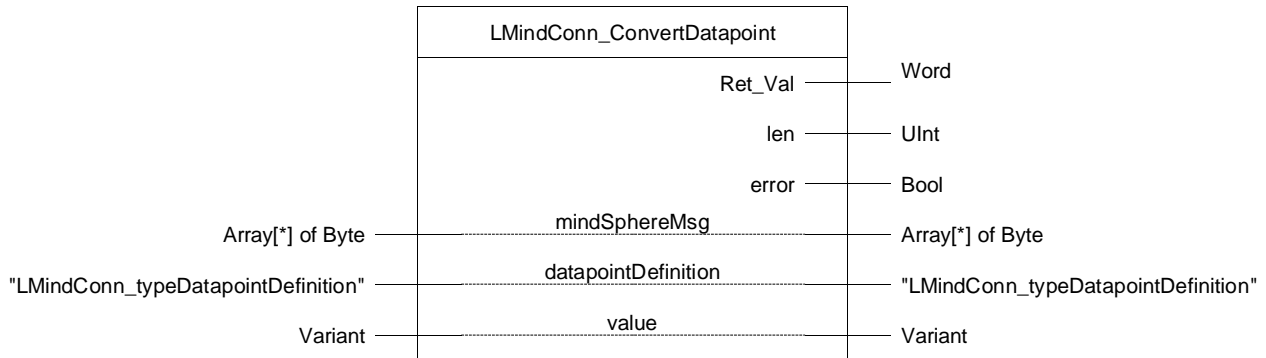


Table 6-6: Parameters of LMindConn\_ConvertDatapoint

Name	Declaration	Data type	Comment
Ret_Val	Return	Word	Status display: <ul style="list-style-type: none"> <li>16#0000: No error</li> <li>16#8201: Data type of the value to be transferred is not supported.</li> <li>16#8202: The definition of the data point is missing or incomplete.</li> <li>16#8203: The array of "mindSphereMsg" is too small for the message. The required length is output at parameter "len".</li> </ul>
len	Output	UInt	Message length In the event of an error stating the "mindSphereMsg" array is not large enough, the required length is output.
error	Output	Bool	TRUE: An error has occurred
mindSphereMsg	InOut	Array[*] of bytes	MindSphere-compliant message
datapointDefinition	InOut	<a href="#">"LMindConn_typeDatapointDefinition"</a>	Definition of data point
value	InOut	Variant	Value to be sent to MindSphere



## 6.4 LMindConn\_ConvertDatapoints50

### Description

The function converts up to 50 data points into a MindSphere-compliant message.

### Parameter

Figure 6-4: LMindConn\_ConvertDatapoints50

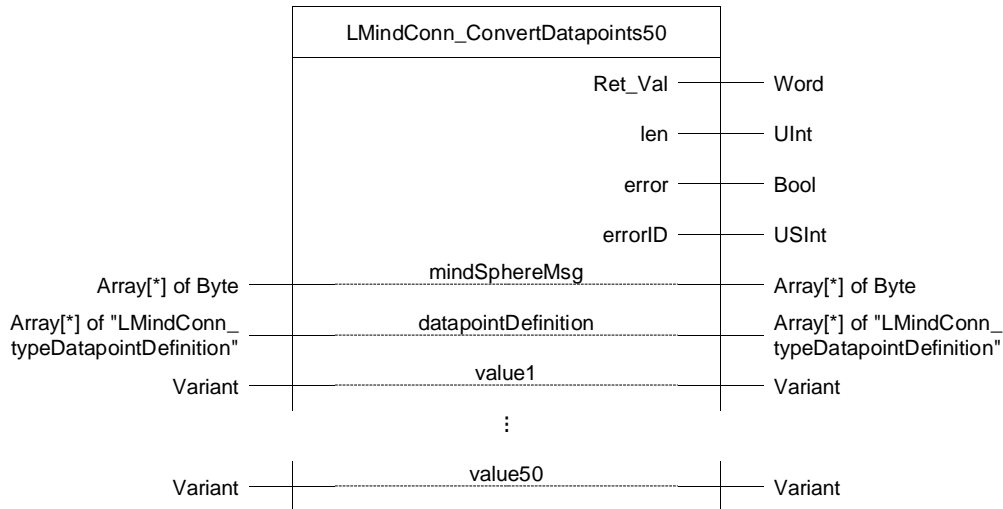


Table 6-7: Parameter of LMindConn\_ConvertDatapoints50

Name	Declaration	Data type	Comment
Ret_Val	Return	Word	Status display: <ul style="list-style-type: none"> <li>16#0000: No error</li> <li>16#8201: Data type of the value to be transferred is not supported.</li> <li>16#8202: The definition of the data point is missing or incomplete.</li> <li>16#8203: The array of "mindSphereMsg" is too small for the message. The required length is output at parameter "len".</li> </ul>
len	Output	UInt	Message length In the event of an error stating the "mindSphereMsg" array is not large enough, the required length is output.
error	Output	Bool	TRUE: An error has occurred
errorID	Output	UInt	ID of the data point (1..50) that caused the error.
mindSphereMsg	InOut	Array[*] of bytes	MindSphere-compliant message
datapointDefinition	InOut	Array[*] of <a href="#">"LMindConn_typeDatapointDefinition"</a>	Definitions of the data points
value1..value50	InOut	Variant	Value to be sent to MindSphere. Switch "NULL" if the value is not used.

#### Note

To transfer more than 50 values to MindSphere, it is necessary to extend the FC.

## 6.5 PLC Data Types

### LMindConn\_typeUserParam

The PLC data type "LMindConn\_typeUserParam" describes the parameters for the connection establishment and the device registration.

Table 6-8: LMindConn\_typeUserParam

Name	Data type	Start value	Description
hwID	HW_ANY	0	Hardware identifier of the PN/IE interface for establishing the connection. If "0" is selected, a suitable one is selected automatically.
connID	CONN_OUC	16#0	Unique connection ID
qdnMCIoTExt	String	"	Full domain name (FQDN) of the MIndConnect IoT Extension The URL must end with a period ("."), e.g., "mciotextension.eu1.mindsphere.io."
enableTls	Bool	TRUE	TRUE: Communication is secured via TLS.
tlsServerCert	UInt	0	ID of the X.509 V3 certificate (usually a CA certificate) to validate the TLS server authentication. If this parameter is "0", the TLS client uses all (CA) certificates currently loaded in the client's certificate store to validate server authentication. Only relevant if "enableTls" equals "TRUE".
username	WString[254]	"	Username for authentication at the MindConnect IoT Extension: <Tenant>/<Username> If empty, the FB with the device ID requests user data from the MindConnect IoT Extension (see Section <a href="#">6.2.2</a> ).
password	WString[32]	"	Password for authentication at the MindConnect IoT Extension If empty, the FB with the device ID requests user data from the MindConnect IoT Extension (see Section <a href="#">6.2.2</a> ).
deviceID	WString[150]	"	Device ID with which the device registers in the MindConnect IoT Extension and requests user data, if necessary
deviceType	WString[25]	"	Device type with which the device registers in the MindConnect IoT Extension
publishQoS	USInt	0	Quality of Service for sending messages: <ul style="list-style-type: none"> <li>• 0: QoS 0</li> <li>• 1: QoS 1</li> <li>• 2: QoS 2</li> </ul>
subscriptionQoS	USInt	0	Quality of Service for receiving messages: <ul style="list-style-type: none"> <li>• 0: QoS 0</li> <li>• 1: QoS 1</li> </ul>

**LMindConn\_typeDatapointDefinition**

This data type contains the definition of a data point.

Table 6-9: LMindConn\_typeDatapointDefinition

Name	Data type	Start value	Description
asset	WString[20]	WSTRING#"	Digital representation of an element, e.g., motor, pump, wind turbine, etc.
type	WString[20]	WSTRING#"	Property of the element, e.g., temperature, power, speed, etc.
unit	WString[5]	WSTRING#"	Measurement unit, e.g., m/s

## 6.6 Integration into the User Project

You will find a detailed application example for the integration of the library into your user project in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/109772284>

# 7 LopcUa

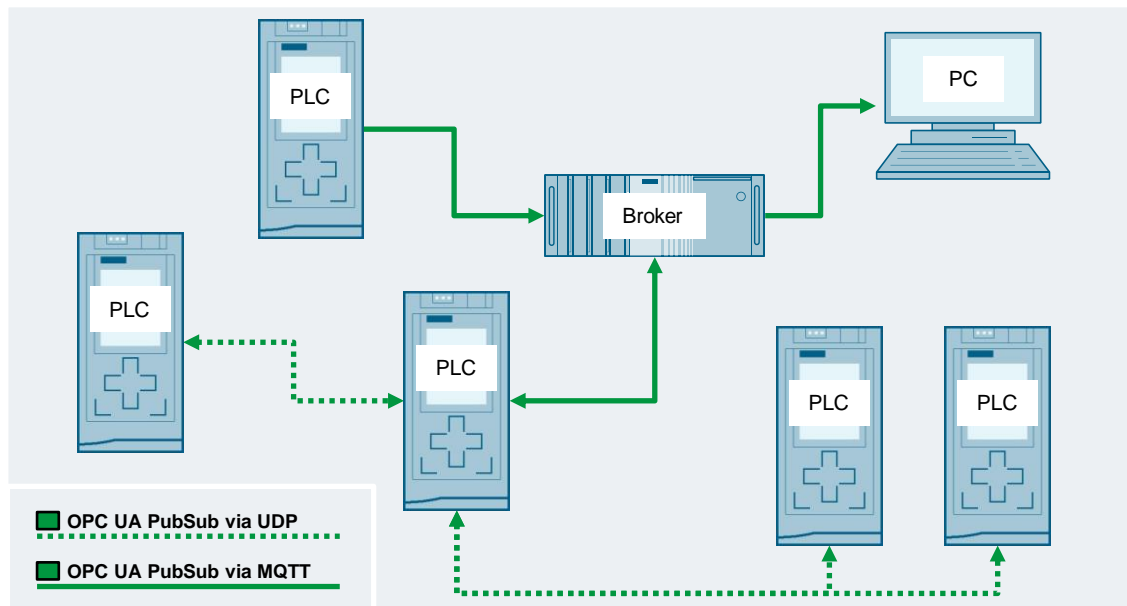
## 7.1 Overview

### 7.1.1 Range of Functions

Part 14 of the OPC UA specification introduced the "PubSub" communication mechanism. This mechanism is not based on the Client–Server principle as in conventional OPC UA communication (OPC 10000-4), but uses a "publish/subscribe" principle.

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. Publishers and Subscribers also do not need to connect directly to each other, allowing not only uni- but also multi- or broadcasts. Additionally the communication is separated from the data storage.

Figure 7-1



The "LopcUa" library provides function blocks that can be used to implement OPC UA PubSub on SIMATIC controllers:

- LopcUa\_PubUdp: Publisher via UDP (UADP encoding)
- LopcUa\_SubUdp: Subscriber via UDP (UADP decoding)
- LopcUa\_PubMqtt: Publisher via MQTT (UADP encoding)
- LopcUa\_SubMqtt: Subscriber via MQTT (UADP decoding)
- LopcUa\_PubMqttJson: Publisher via MQTT (JSON-Encoding)
- LopcUa\_SubMqttJson: Subscriber via MQTT (JSON-Decoding)

**Note**

The blocks "LopcUa\_PubUdp" and "LopcUa\_SubUdp" do not support security.

## 7.1.2 Components of the Library

The "LOpcUa" library contains the following objects.

### Blocks

Table 7-1: Blocks of the library

Name	Type	Version	Description
LOpcUa_PubUdp	FB	V1.3.2	Implements the OPC UA PubSub Publisher via UDP and enables process data to be transmitted to a Subscriber.
LOpcUa_SubUdp	FB	V1.3.1	Implements the OPC UA PubSub Subscriber via UDP and enables process data to be received from a Publisher.
LOpcUa_PubMqtt	FB	V1.2.0	Implements the OPC UA PubSub Publisher via MQTT and enables process data to be transmitted to a Subscriber. Uses "UADP" for encoding.
LOpcUa_SubMqtt	FB	V1.2.0	Implements the OPC UA PubSub Subscriber via MQTT and enables process data to be received from a Publisher. Uses "UADP" for decoding.
LOpcUa_PubMqttJson	FB	V1.0.1	Implements the OPC UA PubSub Publisher via MQTT and enables process data to be transmitted to a Subscriber. Uses "JSON" for encoding.
LOpcUa_SubMqttJson	FB	V1.0.1	Implements the OPC UA PubSub Subscriber via MQTT and enables process data to be received from a Publisher. Uses "JSON" for decoding.
LOpcUa_Pub_Boolean	FC	V1.0.1	Encoding for "LOpcUa_PubX": Boolean to send buffer (byte array).
LOpcUa_Pub_Byte	FC	V1.0.1	Encoding for "LOpcUa_PubX": Byte to send buffer (byte array).
LOpcUa_Pub_Double	FC	V1.0.1	Encoding for "LOpcUa_PubX": Double to send buffer (byte array).
LOpcUa_Pub_Float	FC	V1.0.1	Encoding for "LOpcUa_PubX": Float to send buffer (byte array).
LOpcUa_Pub_Guid	FC	V1.0.1	Encoding for "LOpcUa_PubX": GUID to send buffer (byte array).
LOpcUa_Pub_Int16	FC	V1.0.1	Encoding for "LOpcUa_PubX": Int16 to send buffer (byte array).
LOpcUa_Pub_Int32	FC	V1.0.1	Encoding for "LOpcUa_PubX": Int32 to send buffer (byte array).
LOpcUa_Pub_Int64	FC	V1.0.1	Encoding for "LOpcUa_PubX": Int64 to send buffer (byte array).
LOpcUa_Pub_SByte	FC	V1.0.1	Encoding for "LOpcUa_PubX": SByte to send buffer (byte array).
LOpcUa_Pub_String	FC	V1.1.1	Encoding for "LOpcUa_PubX": String to send buffer (byte array).
LOpcUa_Pub_UInt16	FC	V1.0.1	Encoding for "LOpcUa_PubX": UInt16 to send buffer (byte array).
LOpcUa_Pub_UInt32	FC	V1.0.1	Encoding for "LOpcUa_PubX": UInt32 to send buffer (byte array).
LOpcUa_Pub_UInt64	FC	V1.0.1	Encoding for "LOpcUa_PubX": UInt64 to send buffer (byte array).

## PLC Data Types

Table 7-2: PLC library data types

Name	Version	Description
LOpcUa_typeDataSetWriter	V1.0.0	Maps an OPC UA "DataSetWriter".
LOpcUa_typeDataSetReader	V1.0.3	Maps an OPC UA "DataSetReader".
LOpcUa_typeGuid	V1.0.0	Implements the "GUID" data type.
LOpcUa_typePublishedData	V1.0.0	Implements the send buffer of the Publisher.
LOpcUa_typePublishedDataSet	V1.0.1	Maps an OPC UA "PublishedDataSet". Contains the Publisher process values to be sent.
LOpcUa_typePublisherID	V1.0.4	Maps an OPC UA "PublisherID". Serves to identify the Publisher.
LOpcUa_typeVariant	V1.0.1	Implements the "Variant" data type and maps a "DataSetField".
LOpcUa_typeWriterGroup	V1.0.0	Maps an OPC UA "WriteGroup".
LOpcUa_typeReaderGroup	V1.0.3	Maps an OPC UA "ReaderGroup".
LOpcUa_typeSubscribedDataSet	V1.0.1	Maps an OPC UA "SubscribedDataSet". Contains the received process values of the Publisher.
LOpcUa_typeConnParamMqtt	V1.0.0	Connection parameters for "LOpcUa_PubMqtt" and "LOpcUa_SubMqtt".
LOpcUa_typeDataSetWriter.Json	V1.0.0	Maps an OPC UA "DataSetWriter" for JSON encoding.
LOpcUa_typeDataSetReader.Json	V1.0.0	Maps an OPC UA "DataSetReader" for JSON decoding.
LOpcUa_typePublishedDataSet Json	V1.0.0	Maps an OPC UA "PublishedDataSet" for JSON encoding. Contains the Publisher process values to be sent.
LOpcUa_typeSubscribedDataSet Json	V1.0.0	Maps an OPC UA "SubscribedDataSet" for JSON decoding. Contains the received process values of the Publisher.
LOpcUa_typeWriterGroupJson	V1.0.0	Maps an OPC UA "WriteGroup" for JSON encoding.
LOpcUa_typeReaderGroupJson	V1.0.0	Maps an OPC UA "ReaderGroup" for JSON decoding.

### 7.1.3 Validity

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- CP 1543-1, CP 1542SP-1, CP 1542SP-1 IRC, CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V2.0

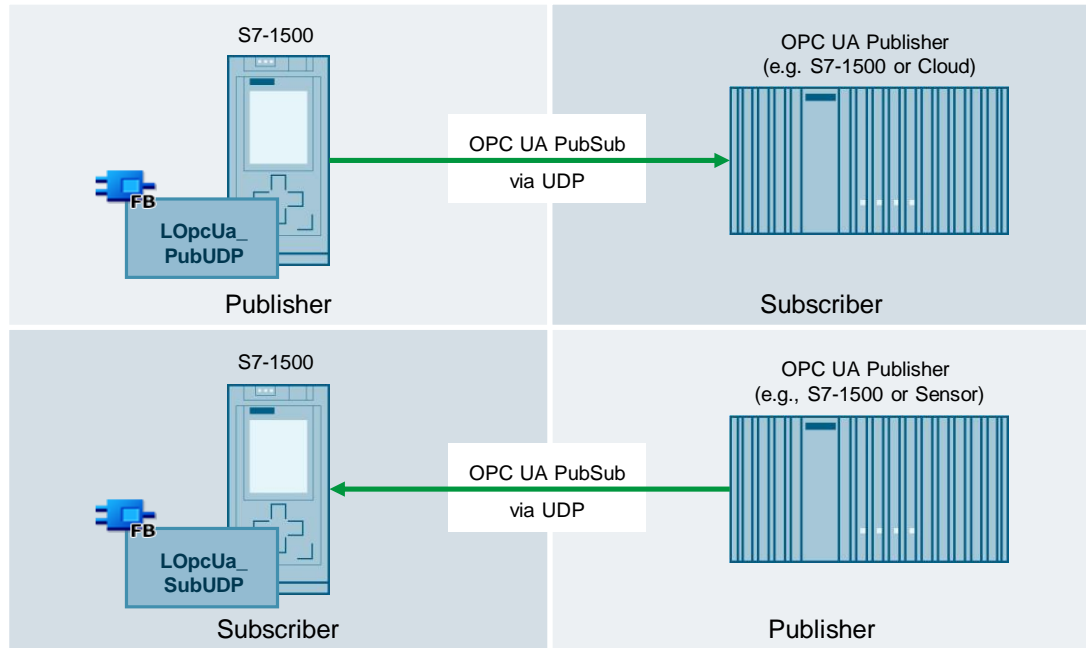
## 7.2 PubSub via UDP

### 7.2.1 Principle of Operation

#### General information

The Publisher is implemented by the function block "LOpcUa\_PubUdp" and the Subscriber by the block "LOpcUa\_SubUdp", which are provided by the library "LOpcUa":

Figure 7-2

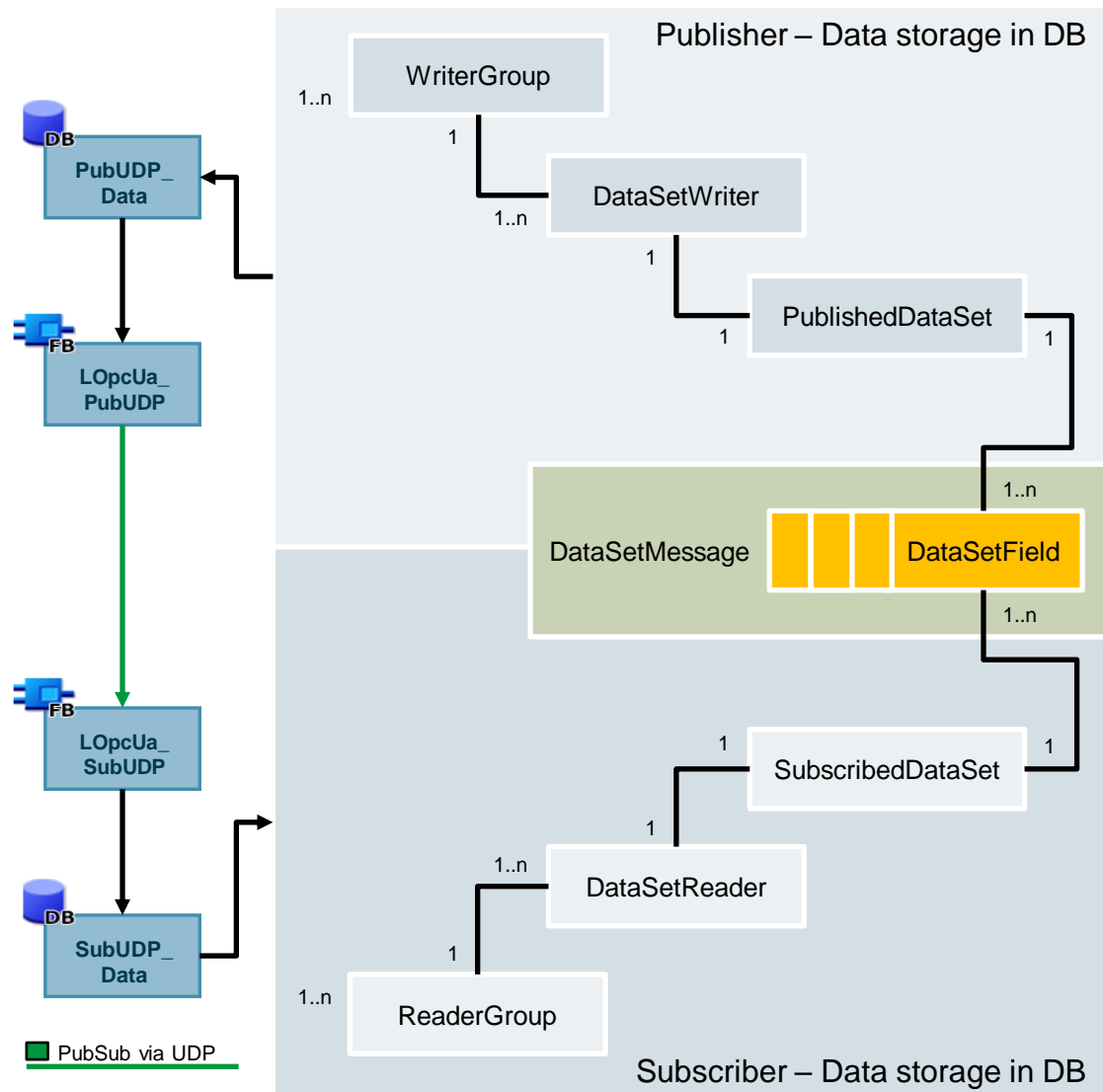


To generate the data storage in a data block, for the process data to be sent or received, the library provides prepared user-defined data types ("UDT"). The Publisher and Subscriber blocks access this data to implement the respective function.

**Schematic representation**

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

Figure 7-3



© Siemens AG 2023. All rights reserved.

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.

To implement the individual components in data blocks, the library provides you with suitable PLC data types that map the required structures. If a component is required more than once ("1..n"), it must be created as an array in the data block.

The block "LOpcUa\_PubUdp" accesses the "WriteGroups" and serializes the "DataSetMessage" from the data contained therein and sends ("Publish") this with the help of the system function block "TUSEND" to the Subscribers.

The block "LOpcUa\_SubUdp" receives the "DataSetMessage" via the system function block "TURCV" and de-serializes the data to the data structure of the created "ReaderGroup".



## 7.2.2 LOPcUa\_PubUdp

### Description

The block "LOpcUa\_PubUdp" sends ("Publish") the PubSub data via UDP to the Subscribers. The block requires a data block in which the components "WriterGroup", "DataSetWriter", "PublishedDataSet", and "PublishedWriterGroup" are predefined.

In an internal sequencer of the block, the data packets are coded and sent via UDP with the help of the system function blocks "TCON", "TUSEND", and "TDISCON".

**Note**

To ensure a consistent send clock, we recommend to call the block in a cyclic interrupt OB. The "PublishingInterval" is defined in the "WriterGroups".

### Parameter

Figure 7-4: LOPcUa\_PubUdp

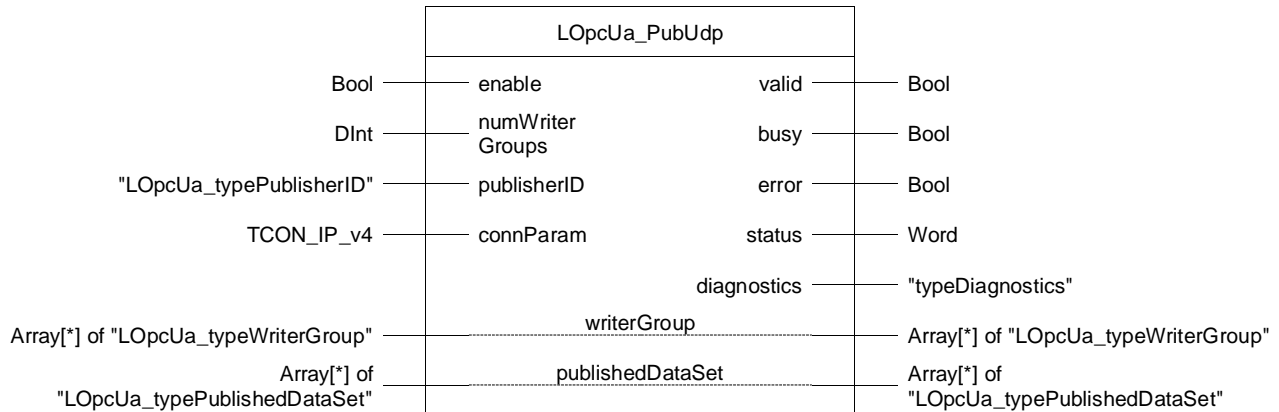


Table 7-3: Parameters of LOPcUa\_PubUdp

Name	Declaration	Data type	Comment
enable	Input	Bool	Control parameters TRUE: Cyclic publishing active. FALSE: Publishing disabled.
numWriterGroups	Input	DInt	Number of "WriteGroups" to be published.
publisherID	Input	WString	"PublisherID" to uniquely identify the publisher.
connParam	Input	TCON_IP_v4	UDP connection parameters.
valid	Output	Bool	TRUE, if publishing is active at the block and no error occurs.
busy	Output	Bool	TRUE, if publishing is active at the block.
error	Output	Bool	TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer. FALSE: No error occurred.
status	Output	Word	Status display Information about the error that occurred in FB "LOpcUa_PubUdp" (see Section 7.2.4).
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section 12.2).
writerGroup	InOut	Array[*] of <a href="#">"LOpcUa_typeWriterGroup"</a>	Data structure for mapping one or more "WriterGroups". Contains metadata for PubSub communication.

Name	Declaration	Data type	Comment
publishedDataSet	InOut	Array[*] of <a href="#">"LOpcUa_type PublishedDataSet"</a>	Data structure for mapping one or more "PublishedDataSets". This structure contains one process tag each, which is published.

### 7.2.3 LOpcUa\_SubUdp

#### Description

The block "LOpcUa\_SubUdp" receives ("Subscribe") the PubSub data via UDP from a Publisher. The block requires a data block in which the "ReaderGroup" component is predefined.

In an internal sequencer of the block, the data packets are received and decoded via the system function blocks "TCON", "TURCV", and "TDISCON" using UDP.

To receive the data with minimum delay we recommend to call the block in a cyclic OB.

#### Parameter

Figure 7-5: LOpcUa\_SubUdp

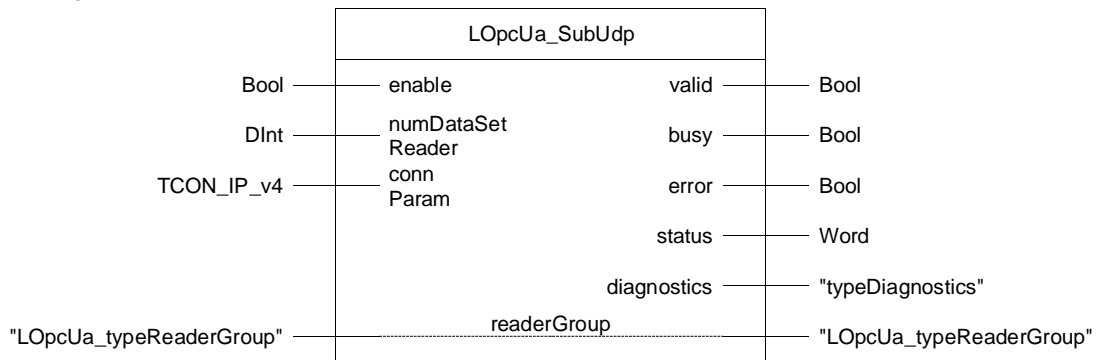


Table 7-4: Parameters of LOpcUa\_SubUdp

Name	Declaration	Data type	Comment
enable	Input	Bool	Control parameters TRUE: Cyclic receiving active. FALSE: Receive disabled.
numDataSetReader	Input	DInt	Number of "DataSetReaders" which should receive the data of the "DataSetWriters" of a Publisher.
connParam	Input	TCON_IP_v4	UDP connection parameters.
valid	Output	Bool	TRUE if subscribing is active on the block and no error occurs.
busy	Output	Bool	TRUE if subscribing is active on the block.
error	Output	Bool	TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer. FALSE: No error occurred.
status	Output	Word	Status display Information about the error that occurred in FB "LOpcUa_SubUdp" (see Section <a href="#">7.2.4</a> ).
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> ).

## 7 LOpcUa

Name	Declaration	Data type	Comment
readerGroup	InOut	<a href="#">"LOpcUa_type ReaderGroup"</a>	Data structure for mapping a "ReaderGroup". Contains the structures of the "DataSetReader".

### Note

The block "LOpcUa\_SubUdp" supports only one "ReaderGroup".

### 7.2.4 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The status and error codes specific to the blocks "LOpcUa\_PubUdp" and "LOpcUa\_SubUdp" are listed below.

Table 7-5: Block output "status"

Code	Description
16#7000	No error; block ready for execution.
16#7002	No error; publishing/subscribing via UDP active.
16#8600	Configuration error: Check whether enough "WriterGroups", related to the block input "numWriterGroups", have been created. In addition, check whether the fields of the structures are assigned correctly.
16#8601	Connection error for TCON: For more information, refer to the "diagnostics" block output.
16#8602	Receive error for TURCV: For more information, refer to the "diagnostics" block output.
16#8603	Send error for TUSEND: For more information, refer to the "diagnostics" block output.
16#8604	Error while encoding the data: Check the metadata on the "writerGroup" parameter.
16#8605	Error decoding the data: Received data type is not supported. Extend the function block or use a different data type with the Publisher.

### 7.2.5 Integration into the User Project

You will find a detailed application example for the integration of the library into your user project and further information about OPC UA PubSub in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/109782455>

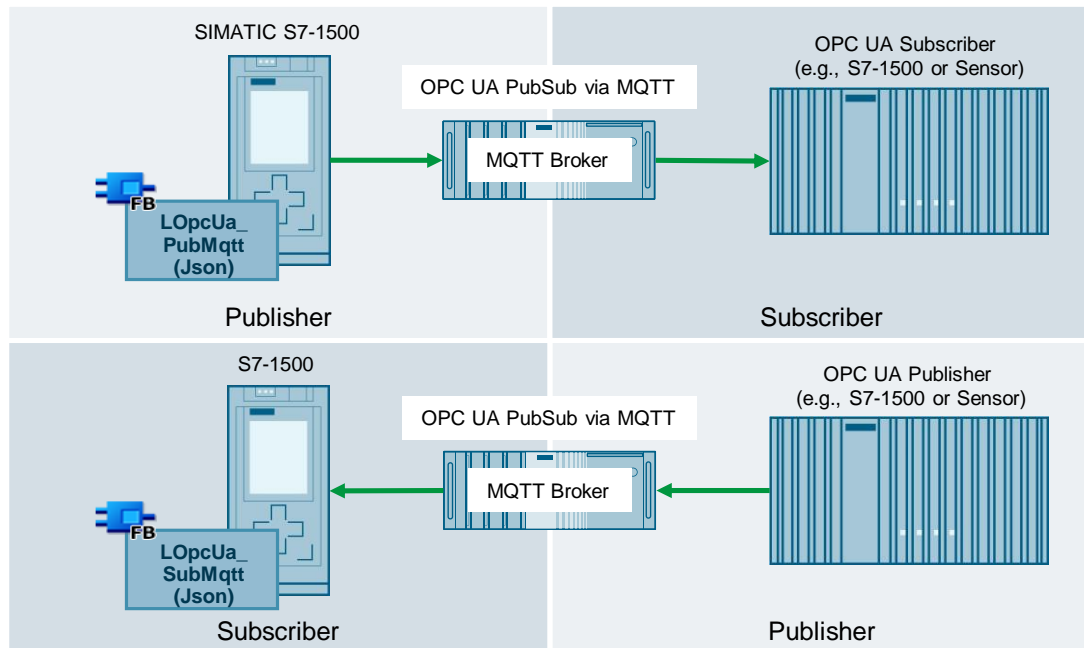
## 7.3 PubSub via MQTT

### 7.3.1 Principle of Operation

#### General 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":

Figure 7-6

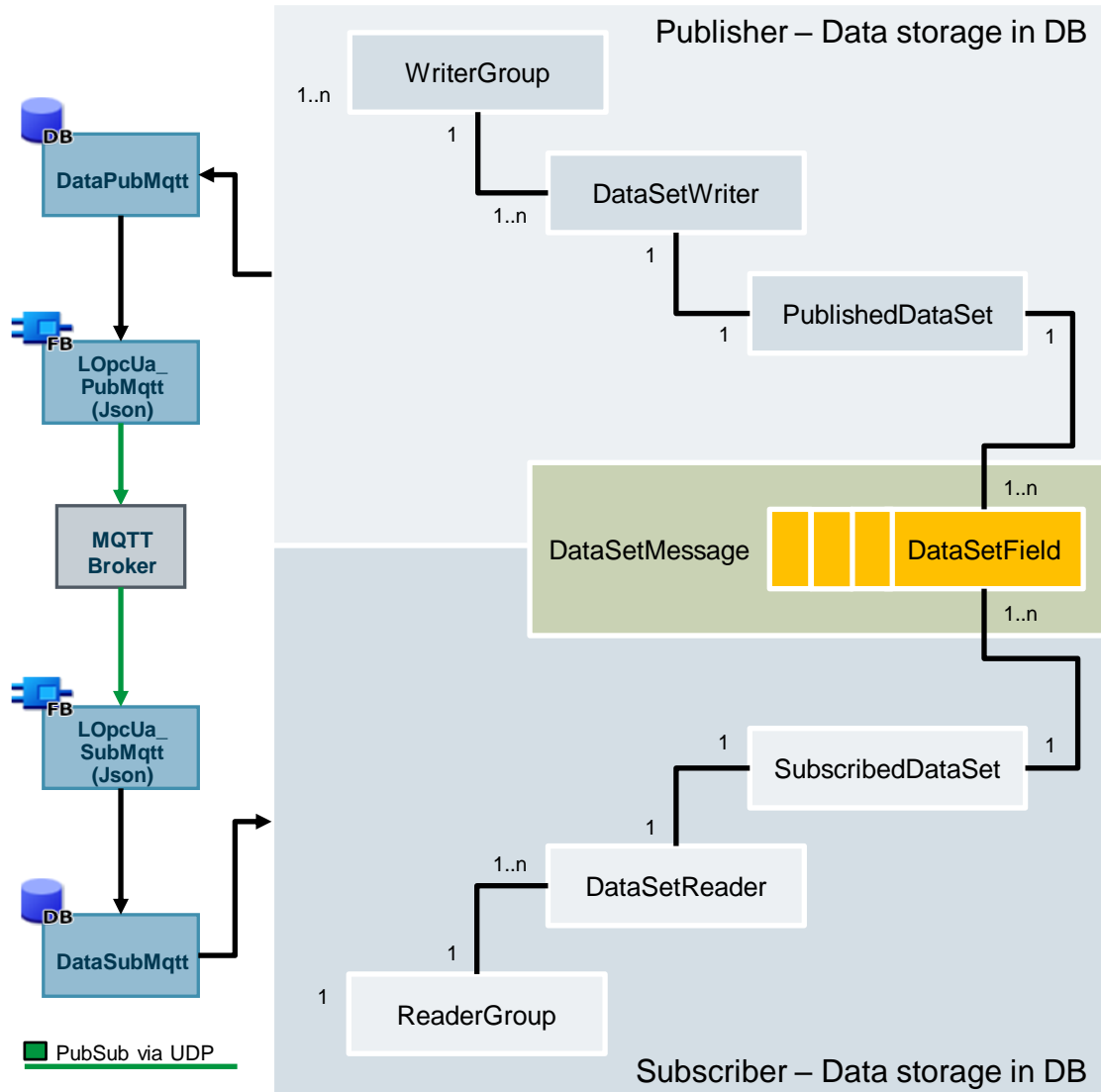


To generate the data storage in a data block, for the process data to be sent or received, the library provides prepared user-defined data types ("UDT"). The Publisher and Subscriber blocks access this data store in order to implement the respective function. Depending on the chosen block the encoding is done via UADP or JSON.

**Schematic representation**

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

Figure 7-7



© Siemens AG 2023. All rights reserved.

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.

To implement the individual components in data blocks, the library provides you with suitable PLC data types that map the required structures. If a component is required more than once ("1..n"), it must be created as an array in the data block.

The block "LOpcUa\_PubUdp" accesses the "WriteGroups" and serializes the "DataSetMessage" from the data contained therein and sends ("Publish") this with the help of the system function block "TUSEND" to the Subscribers.

The block "LOpcUa\_SubUdp" receives the "DataSetMessage" via the system function block "TURCV" and de-serializes the data to the data structure of the created "ReaderGroup".

### 7.3.2 LopcUa\_PubMqtt

#### Description

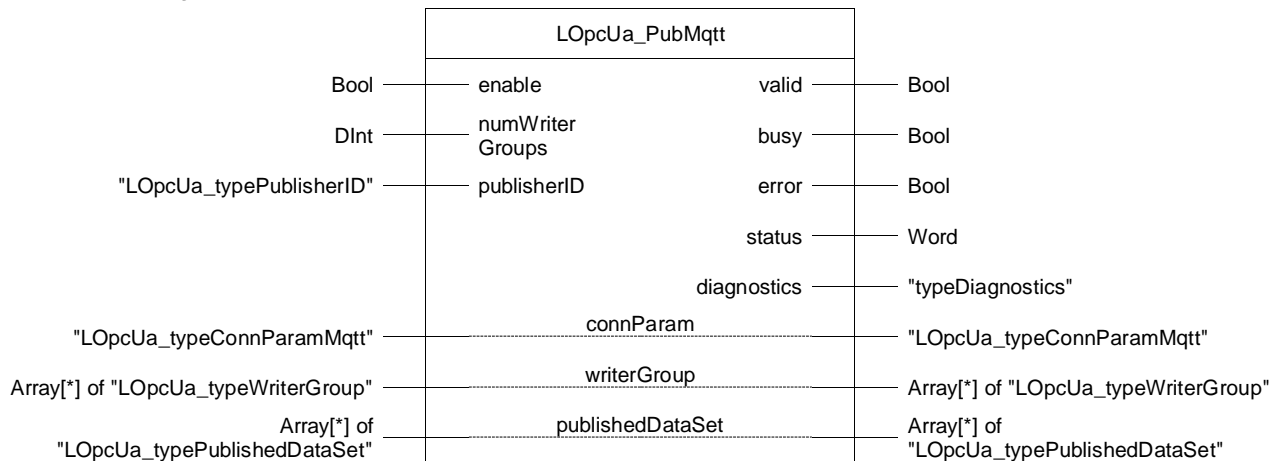
The block "LopcUa\_PubMqtt" sends ("Publish") the PubSub data via MQTT to an MQTT Broker. The block requires a data block in which the components "WriterGroup", "DataSetWriter", "PublishedDataSet", and "PublishedWriterGroup" are predefined.

In an internal step chain of the block, the data packets are coded and sent via MQTT using the library block "[LMQTT\\_Client](#)" is used to send them via MQTT.

**Note** To ensure a consistent send clock, we recommend to call the block in a cyclic interrupt OB. The "PublishingInterval" is defined in the "WriterGroups".

#### Parameter

Figure 7-8: LopcUa\_PubMqtt



© Siemens AG 2023. All rights reserved.

Table 7-6: Parameter of LopcUa\_PubMqtt

Name	Declaration	Data type	Comment
enable	Input	Bool	Control parameters TRUE: Cyclic publishing active. FALSE: Publishing disabled.
numWriterGroups	Input	DInt	Number of "WriteGroups" to be published.
publisherID	Input	<a href="#">"LopcUa_typePublisherID"</a>	"PublisherID" to uniquely identify the publisher.
valid	Output	Bool	TRUE, if publishing is active at the block and no error occurs.
busy	Output	Bool	TRUE, if publishing is active at the block.
error	Output	Bool	TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer. FALSE: No error occurred.
status	Output	Word	Status display Information about the error that occurred in FB "LopcUa_PubUdp" (see Section <a href="#">7.3.6</a> ).
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> ).

Name	Declaration	Data type	Comment
connParam	InOut	<a href="#">"LOpcUa_type ConnParamMqtt"</a>	MQTT connection parameters.
writerGroup	InOut	Array[*] of <a href="#">"LOpcUa_type WriterGroup"</a>	Data structure for mapping one or more "WriterGroups". Contains metadata for PubSub communication.
publishedDataSet	InOut	Array[*] of <a href="#">"LOpcUa_type PublishedDataSet"</a>	Data structure for mapping one or more "PublishedDataSets". This structure contains one process tag each, which is published.

### 7.3.3 LOpcUa\_SubMqtt

#### Description

The block "LOpcUa\_SubMqtt" receives ("Subscribe") the PubSub data via MQTT from an MQTT Broker. The block requires a data block in which the "ReaderGroup" component is predefined.

In an internal step chain of the block, the data packets are received and decoded via MQTT using the library block "[LMQTT\\_Client](#)" library module to receive and decode the data packets via MQTT.

To receive the data with minimum delay we recommend to call the block in a cyclic OB.

#### Parameter

Figure 7-9: LOpcUa\_SubMqtt

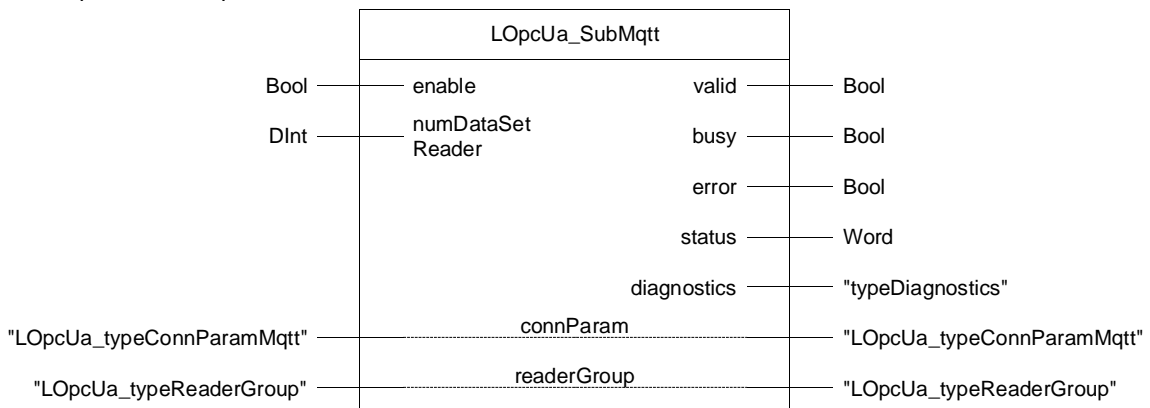


Table 7-7: Parameter of LOpcUa\_SubMqtt

Name	Declaration	Data type	Comment
enable	Input	Bool	Control parameters TRUE: Cyclic receiving active. FALSE: Receive disabled.
numDataSetReader	Input	DInt	Number of "DataSetReaders" which should receive the data of the "DataSetWriters" of a Publisher.
valid	Output	Bool	TRUE if subscribing is active on the block and no error occurs.
busy	Output	Bool	TRUE if subscribing is active on the block.
error	Output	Bool	TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer. FALSE: No error occurred.

## 7 LopcUa

Name	Declaration	Data type	Comment
status	Output	Word	Status display Information about the error that occurred in FB "LopcUa_SubUdp" (see Section <a href="#">7.3.6</a> ).
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> ).
connParam	InOut	<a href="#">"LopcUa_type ConnParamMqtt"</a>	MQTT connection parameters.
readerGroup	InOut	<a href="#">"LopcUa_type ReaderGroup"</a>	Data structure for mapping a "ReaderGroup". Contains the structures of the "DataSetReader".

**Note**

The block "LopcUa\_SubMqtt" supports only one "ReaderGroup".



### 7.3.4 LopcUa\_PubMqttJson

#### Description

The block "LopcUa\_PubMqttJson" sends ("Publish") the PubSub data via MQTT to an MQTT Broker. The block requires a data block in which the components "WriterGroup", "DataSetWriter", "PublishedDataSet", and "PublishedWriterGroup" are predefined.

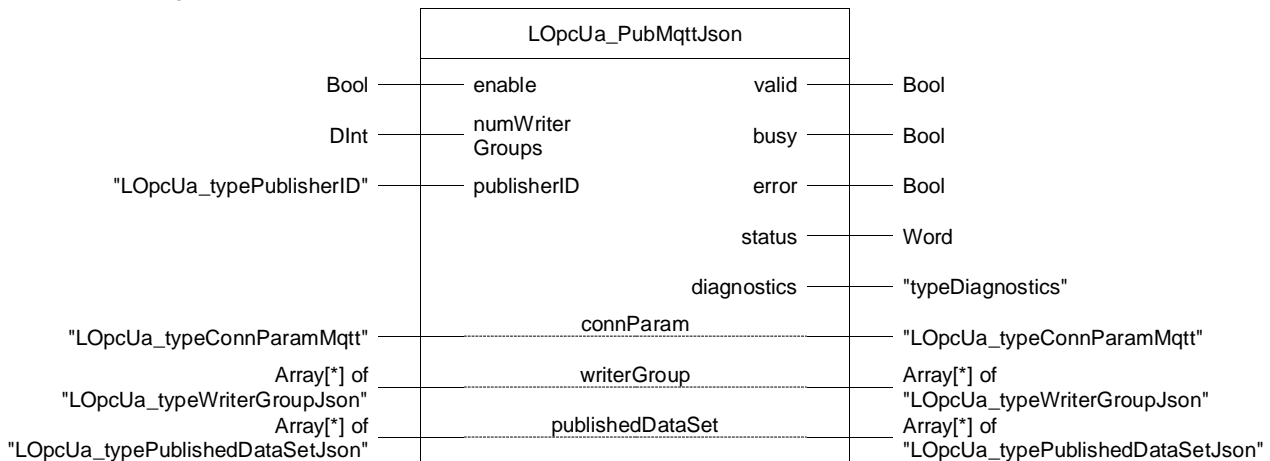
In an internal step chain of the block, the data packets are coded and sent via MQTT using the library block "[LMQTT\\_Client](#)" is used to send them via MQTT.

**Note**

To ensure a consistent send clock, we recommend to call the block in a cyclic interrupt OB. The "PublishingInterval" is defined in the "WriterGroups".

#### Parameter

Figure 7-10: LopcUa\_PubMqttJson



© Siemens AG 2023. All rights reserved.

Table 7-8: Parameter of LopcUa\_PubMqtt

Name	Declaration	Data type	Comment
enable	Input	Bool	Control parameters TRUE: Cyclic publishing active. FALSE: Publishing disabled.
numWriterGroups	Input	DInt	Number of "WriteGroups" to be published.
publisherID	Input	WString	"PublisherID" to uniquely identify the publisher.
valid	Output	Bool	TRUE, if publishing is active at the block and no error occurs.
busy	Output	Bool	TRUE, if publishing is active at the block.
error	Output	Bool	TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer. FALSE: No error occurred.
status	Output	Word	Status display Information about the error that occurred in FB "LopcUa_PubUdp" (see Section <a href="#">7.3.6</a> ).
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> ).

## 7 LOpcUa

Name	Declaration	Data type	Comment
connParam	InOut	<a href="#">"LOpcUa_typeConnParamMqtt"</a>	MQTT connection parameters.
writerGroup	InOut	Array[*] of <a href="#">"LOpcUa_typeWriterGroupJson"</a>	Data structure for mapping one or more "WriterGroups". Contains metadata for PubSub communication.
publishedDataSet	InOut	Array[*] of <a href="#">"LOpcUa_typePublishedDataSetJson"</a>	Data structure for mapping one or more "PublishedDataSets". This structure contains one process tag each, which is published.

### 7.3.5 LOpcUa\_SubMqttJson

#### Description

The block "LOpcUa\_SubMqttJson" receives ("Subscribe") the PubSub data via MQTT from an MQTT Broker. The block requires a data block in which the "ReaderGroup" component is predefined.

In an internal step chain of the block, the data packets are received and decoded via MQTT using the library block "[LMQTT\\_Client](#)" library module to receive and decode the data packets via MQTT.

To receive the data with minimum delay we recommend to call the block in a cyclic OB.

#### Parameter

Figure 7-11: LOpcUa\_SubMqttJson

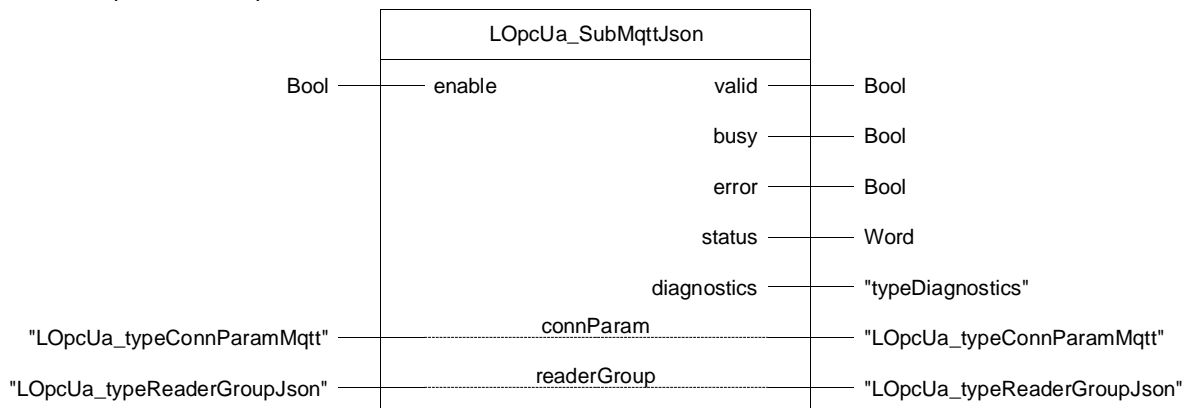


Table 7-9: Parameter of LOpcUa\_SubMqtt

Name	Declaration	Data type	Comment
enable	Input	Bool	Control parameters TRUE: Cyclic receiving active. FALSE: Receive disabled.
valid	Output	Bool	TRUE if subscribing is active on the block and no error occurs.
busy	Output	Bool	TRUE if subscribing is active on the block.
error	Output	Bool	TRUE: An error has occurred. More detailed information is provided by the output parameter "status" or the diagnostic buffer. FALSE: No error occurred.
status	Output	Word	Status display Information about the error that occurred in FB "LOpcUa_SubUdp" (see Section <a href="#">7.3.6</a> ).

## 7 LOPcUa

Name	Declaration	Data type	Comment
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> ).
connParam	InOut	<a href="#">"LOpcUa_typeConnParamMqtt"</a>	MQTT connection parameters.
readerGroup	InOut	<a href="#">"LOpcUa_typeReaderGroupJson"</a>	Data structure for mapping a "ReaderGroup". Contains the structures of the "DataSetReader".

### Note

The block "LOpcUa\_SubMqttJson" supports only one "ReaderGroup".

### 7.3.6 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The status and error codes specific to the blocks "LOpcUa\_PubMqtt" and "LOpcUa\_SubMqtt" are listed below.

Table 7-10: Block output "status"

Code	Description
16#7000	No error; block ready for execution.
16#7002	No error; publishing/subscribing via UDP active.
16#8600	Configuration error: Check whether enough "WriterGroups", related to the block input "numWriterGroups", have been created. In addition, check whether the fields of the structures are assigned correctly.
16#8601	Connection error for "LMQTT_Client": For more information, refer to the "diagnostics" block output or check the "Status" parameter in the "instLMQTT_Client" multi-instance.
16#8602	Receive error for "LMQTT_Client": For more information, refer to the "diagnostics" block output or check the "Status" parameter in the "instLMQTT_Client" multi-instance.
16#8603	Send error for "LMQTT_Client": For more information, refer to the "diagnostics" block output or check the "Status" parameter in the "instLMQTT_Client" multi-instance.
16#8604	Error while encoding the data: Check the metadata on the "writerGroup" parameter.
16#8605	Error decoding the data: Received data type is not supported. Extend the function block or use a different data type with the Publisher.

### 7.3.7 Integration into the User Project

You will find a detailed application example for the integration of the library into your user project and further information about OPC UA PubSub in the Siemens Industry Online Support:

- MQTT with UADP-Encoding  
<https://support.industry.siemens.com/cs/ww/en/view/109797826>
- MQTT with JSON-Encoding  
<https://support.industry.siemens.com/cs/ww/en/view/109814033>

## 7.4 PLC Data Types

The following descriptions explain the PLC data types that map the PubSub components for communication.

### LOpcUa\_typePublisherID

The PLC data type "LOpcUa\_typePublisherID" is used to identify the Publisher.

Table 7-11: Parameters of "LOpcUa\_typePublisherID"

Name	Data type	Description
type	UDInt	Selects the data type to be used for the Publisher ID to be sent: <ul style="list-style-type: none"> <li>"3": id8</li> <li>"5": id16</li> <li>"7": id32</li> <li>"9": id64</li> <li>"12": idString</li> </ul>
id8	USInt	The tag selected with "type" contains the Publisher ID that identifies your Publisher. You can choose the ID freely. Example: "type" = "7" corresponds to "id32 (UDInt)"
id16	UInt	
id32	UDInt	
id64	ULInt	
idString	WString	

### LOpcUa\_typeWriterGroup

The PLC data type "LOpcUa\_typeWriterGroup" implements the WriterGroups.

Table 7-12: Parameters of "LOpcUa\_typeWriterGroup"

Name	Data type	Description
name	String	The name of the "WriterGroup" (optional, will not be transferred).
priority	USInt	The priority for sorting (higher value = higher priority).
publishingInterval	USInt	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	USInt	Must contain the same value as the "PublishingInterval".
numDataSetWriter	USInt	Number of "DataSetWriters" you want to use.
dataSetWriter	Array [0..X] of "LOpcUa_typeDataSetWriter"	Implements the "DataSetWriter" per "WriterGroup".

### LOpcUa\_typeDataSetWriter

The PLC data type "LOpcUa\_typeDataSetWriter" implements the DataSetWriter.

Table 7-13: Parameters of "LOpcUa\_typeDataSetWriter"

Name	Data type	Description
name	String	The name of the "DataSetWriter" (optional, will not be transferred).
id	UInt	Identification number of the "DataSetWriter" (any).
sequenceNumber	UInt	Is not filled in (required by FB "LOpcUa_PubUdp").

Name	Data type	Description
dataSet	UInt	Reference to the index of the "PublishedDataSet". Each "DataSet" refers to a different index of the "PublishedDataSets".

**LOpcUa\_typePublishedDataSet**

The PLC data type "LOpcUa\_typePublishedDataSet" implements the PublishedDataSets.

Table 7-14: Parameters of "LOpcUa\_typePublishedDataSet"

Name	Data type	Description
name	String	The name of the "PublishedDataSet" (optional, will not be transmitted).
fieldCount	UInt	Number of "DataSetFields" in the "PublishedDataSet" to be published.
dataSetField	Array [0..X] of "LOpcUa_typeVariant"	Implements the "DataSetFields" which contain the individual process tags to be sent.

**LOpcUa\_typeVariant**

The PLC data type "LOpcUa\_typeVariant" implements the data type "Variant" for PubSub communication.

Table 7-15: Parameters of "LOpcUa\_typeVariant"

Name	Data type	Description
encodingMask	USInt	Specifies the data type of the process value to be sent or received.
value	Dependent on "encodingMask"	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"

**LOpcUa\_typeReaderGroup**

The PLC data type "LOpcUa\_typeReaderGroup" implements the ReaderGroup.

Table 7-16: Parameters of "LOpcUa\_typeReaderGroup"

Name	Data type	Description
name	String	The name of the "ReaderGroup" (optional).
dataSetReader	Array [0..X] of "LOpcUa_typeDataSetReader"	Implements the "DataSetReader" per "ReaderGroup".

**LopcUa\_typeDataSetReader**

The PLC data type "LopcUa\_typeDataSetReader" implements the DataSetReader.

Table 7-17: Parameters of "LopcUa\_typeDataSetReader"

Name	Data type	Description
name	String	The name of the "DataSetReader" (optional).
publisherID	"LopcUa_type PublisherID"	The received PublisherID.
dataSetWriterID	UInt	The received DataSetWriterID.
sequenceNumber	UInt	Sequence number of the received PubSub packets
subscribedDataSet	"LopcUa_type SubscribedDataSet"	Contains the "DataSetFields" of type "LopcUa_typeVariant" in which the received process values are stored.

**LopcUa\_typeSubscribedDataSet**

The PLC data type "LopcUa\_typeSubscribedDataSet" implements the SubscribedDataSets.

Table 7-18: Parameters of "LopcUa\_typeSubscribedDataSet"

Name	Data type	Description
name	String	The name of the SubscribedDataSet" (optional, will not be transmitted).
fieldCount	UInt	Number of "DataSetFields" in the "PublishedDataSet" to be published.
dataSetField	Array [0..X] of "LopcUa_typeVariant"	Implements the "DataSetFields" which contain the individual process tags to be sent.

**LopcUa\_typeWriterGroupJson**

The PLC data type "LopcUa\_typeWriterGroupJson" implements the WriterGroups.

Table 7-19: Parameters of "LopcUa\_typeWriterGroupJson"

Name	Data type	Description
name	String	The name of the "WriterGroup" (optional, will not be transferred).
priority	USInt	The priority for sorting (higher value = higher priority).
publishingInterval	USInt	The publishing interval for the configuration of the send clock. The transmission rate is given by the following formula: <i>Send clock = OB cycle time * PublishingInterval</i>
resetInterval	USInt	Must contain the same value as the "PublishingInterval".
numDataSetWriter	USInt	Number of "DataSetWriters" you want to use.
dataSetWriter	Array [0..X] of "LopcUa_typeDataSetWriterJson"	Implements the "DataSetWriter" per "WriterGroup".

**LopcUa\_typeDataSetWriterJson**

The PLC data type "LopcUa\_typeDataSetWriterJson" implements the DataSetWriter.

Table 7-20: Parameters of "LopcUa\_typeDataSetWriterJson"

Name	Data type	Description
id	WString	Identification number of the "DataSetWriter" (any).
sequenceNumber	UInt	Is not filled in (required by FB "LopcUa_PubUdpJson").
dataSet	UInt	Reference to the index of the "PublishedDataSet". Each "DataSet" refers to a different index of the "PublishedDataSets".

**LopcUa\_typePublishedDataSetJson**

The PLC data type "LopcUa\_typePublishedDataSetJson" implements the PublishedDataSets.

Table 7-21: Parameters of "LopcUa\_typePublishedDataSetJson"

Name	Data type	Description
name	WString	The name of the "PublishedDataSet".
fieldCount	UInt	Number of "DataSetFields" in the "PublishedDataSet" to be published.
dataSetField	Array [0..X] of "LopcUa_typeVariant"	Implements the "DataSetFields" which contain the individual process tags to be sent.

**LopcUa\_typeReaderGroupJson**

The PLC data type "LopcUa\_typeReaderGroupJson" implements the ReaderGroup.

Table 7-22: Parameters of "LopcUa\_typeReaderGroupJson"

Name	Data type	Description
name	String	The name of the "ReaderGroup" (optional).
dataSetReader	Array [0..X] of "LopcUa_typeDataSetReaderJson"	Implements the "DataSetReader" per "ReaderGroup".

**LopcUa\_typeDataSetReaderJson**

The PLC data type "LopcUa\_typeDataSetReaderJson" implements the DataSetReader.

Table 7-23: Parameters of "LopcUa\_typeDataSetReaderJson"

Name	Data type	Description
name	String	The name of the "DataSetReader" (optional).
publisherID	WString	The received PublisherID.
dataSetWriterID	WString	The received DataSetWriterID.
sequenceNumber	UInt	Sequence number of the received PubSub packets
subscribedDataSet	"LopcUa_typeSubscribedDataSetJson"	Contains the "DataSetFields" of type "LopcUa_typeVariant" in which the received process values are stored.

**LopcUa\_typeSubscribedDataSetJson**

The PLC data type "LopcUa\_typeSubscribedDataSetJson" implements the SubscribedDataSets.

Table 7-24: Parameters of "LopcUa\_typeSubscribedDataSetJson"

Name	Data type	Description
name	WString	The name of the SubscribedDataSet" (optional, will not be transmitted).
fieldCount	UInt	Number of "DataSetFields" in the "PublishedDataSet" to be published.
dataSetField	Array [0..X] of "LopcUa_typeVariant"	Implements the "DataSetFields" which contain the individual process tags to be sent.

**LopcUa\_typeConnParamMqtt**

The PLC data type "LopcUa\_typeConnParamMqtt" defines the connection parameters for the MQTT connection.

Table 7-25: Parameters of "LopcUa\_typeConnParamMqtt"

Name	Data type	Description
mqttConnParam	<a href="#">"LMQTT_typeConnParam"</a>	Basic MQTT connection parameters.
mqttClientIdentifier	WString	MQTT Client identifier used when establishing the connection.
mqttUserName	WString	Optional: Username for connection establishment.
mqttPassword	WString	Optional: Password for connection establishment.
mqttTopic	WString	MQTT topic used for publish/subscribe.



## 8 LSNMP

### 8.1 Overview

#### 8.1.1 Range of Functions

The status of SNMP-capable network components can be monitored and, if necessary, controlled via SNMP (Simple Network Management Protocol) by network management systems, such as SINEC NMS.

The blocks of the "LSNMP" library also allow a SIMATIC Controller with a PROFINET interface, acting as a simple SNMP manager, to query information from the network components and, if necessary, to control them as well, or to send SNMP traps to an SNMP manager as an SNMP agent.

The following functions are implemented in the library:

- Send SNMP GetRequests and output response
- Send SNMP GetNextRequests and output response
- Send SNMP GetBulkRequests and output response
- Send SNMP SetRequests
- Send SNMP traps

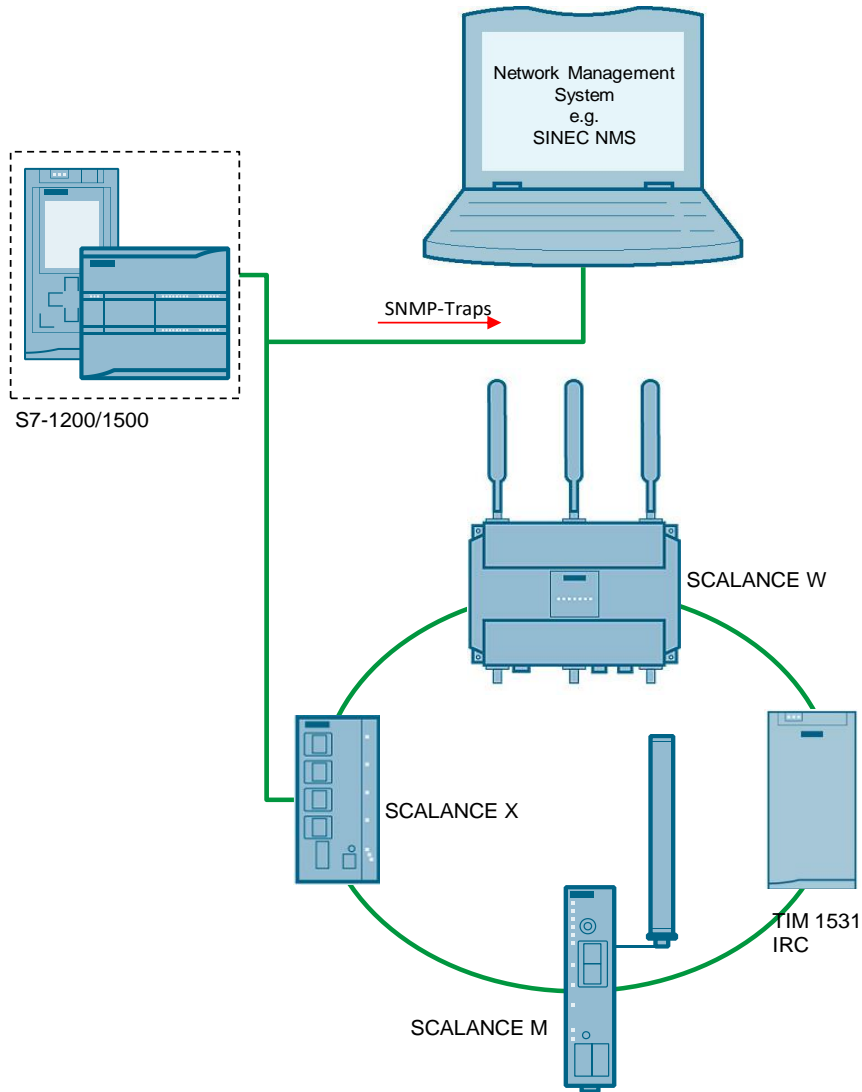
**Note**

The blocks of this library support sending and receiving SNMP messages that do not exceed 486 bytes in total length.

**Diagram**

The following figure shows a possible constellation in which you can use the SNMP blocks of the "LSNMP" library.

Figure 8-1



## 8.1.2 Components of the Library

The "LSNMP" library contains the following objects:

### Blocks

Table 8-1: Blocks of the library

Name	Type	Version	Description
LSNMP_Get	FB	V5.0.0	Request of an SNMP tag from an SNMP agent (SNMPv1) (GetRequest or GetNextRequest)
LSNMP_GetBulk	FB	V5.0.0	Requesting large amounts of data from an SNMP agent with only a single response telegram (SNMPv2c)
LSNMP_Set	FB	V5.0.0	Changing an SNMP tag of an SNMP agent (SNMPv1)
LSNMP_SendTrap	FB	V5.0.0	Sends user-defined traps to an SNMP manager (SNMPv1)
LSNMP_CalcCnt	FC	V1.0.0	Used to calculate the number of bytes a BER encoded length takes in the packet (not relevant for the user)
LSNMP_DecodeLen	FC	V1.0.0	Decodes a BER encoded length (not relevant for the user)
LSNMP_EncodeLen	FC	V1.0.0	Encodes a length according to BER and inserts it into the packet (not relevant for the user)

### PLC Data Types

Table 8-2: PLC data types of the library

Name	Version	Description
LSNMP_typeConnParam	V1.0.0	Describes the connection parameters for all LSNMP blocks
LSNMP_typeDataSendTrap	V5.0.0	Describes the data that is sent with "LSNMP_SendTrap"
LSNMP_typePacket	V1.0.0	Structures parameters that are necessary for building the SNMP package within the blocks (not relevant for the user)
LSNMP_typeVarBind	V1.0.0	Describes a tag binding to be sent or received

## 8.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.2
- SIMATIC ET 200SP Open Controllers as of firmware V2.0
- SIMATIC S7-1500 Software Controllers as of Firmware V2.0
- CP 1543-1, CP 1542SP-1, CP 1543SP-1 IRC, CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V2.0
- SNMPv1/SNMPv2c

## 8.2 LSNMP\_Get

### Description

SNMP provides different data packets to request tags of an SNMP agent. Two of them are implemented in this block:

- GetRequest to request a specific tag
- GetNextRequest to request the next tag in the MIB tree

For this purpose, the block sends either an SNMP GetRequest or SNMP GetNextRequest command for a tag by means of an Object Identifier (OID) to an SNMP agent. The SNMP agent responds with an SNMP GetResponse telegram containing the requested data or an error message.

### Parameter

Figure 8-2: LSNMP\_Get

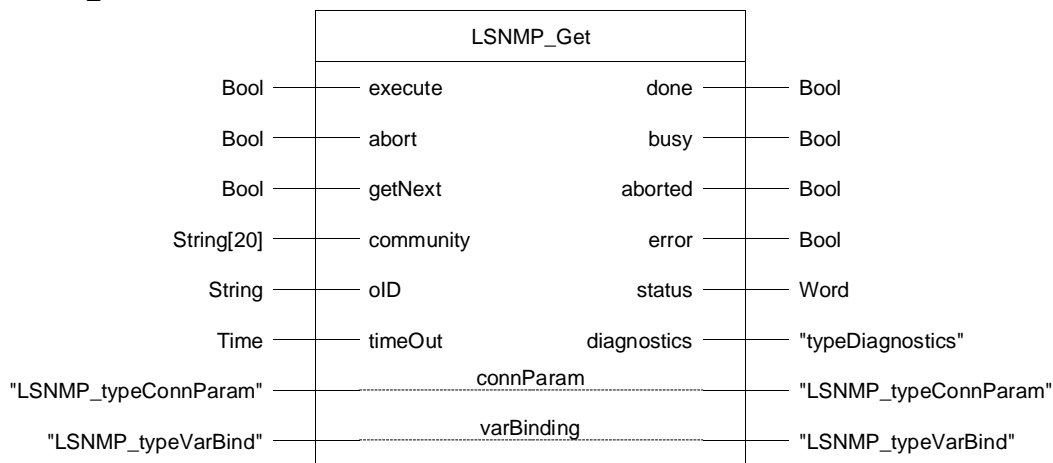


Table 8-3: Parameters of LSNMP\_Get

Name	Declaration	Data type	Comment
execute	Input	Bool	The job is started with a positive edge. "abort" must be FALSE.
abort	Input	Bool	With a positive edge, the current job is canceled. The command is only accepted if the block is currently busy.
getNext	Input	Bool	FALSE: Requests the tag for the specified OID TRUE: Requests the next tag in the MIB tree for the specified OID
community	Input	String[20]	Community
oid	Input	String	Object Identifier of the requested tag according to dot notation, e.g. '1.3.6.1.2.1.1.5.0'
timeOut	Input	Time	Time after which a job should be automatically canceled
done	Output	Bool	TRUE: Job successfully executed. The received data is available at the "varBinding" parameter.

Name	Declaration	Data type	Comment
busy	Output	Bool	TRUE: Job is being executed
aborted	Output	Bool	TRUE: Job was canceled
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section 8.8)
diagnostics	Output	"typeDiagnostics"	Advanced diagnostic information (see Section 12.2)
connParam	InOut	"LSNMP_typeConnParam"	Connection parameters
varBinding	InOut	"LSNMP_typeVarBind"	Received variable binding

### 8.3 LSNMP\_GetBulk

#### Functional description

The block "LSNMP\_GetBulk" is used to read out large amounts of data with only one response telegram.

The block "LSnmp\_GetBulk" uses the SNMP GetBulk request command and requires SNMPv2 for this – an extension of SNMPv1.

The GetBulk command performs several GetNext queries in the SNMP agent. The maximum number of GetNext commands can be specified via a parameterizable repetition factor in the GetBulk telegram.

The return values of all queried objects are compiled in a single response telegram and transferred. The block "LSNMP\_GetBulk" then outputs the received data stream divided into the individual tags.

#### Parameter

Figure 8-3: LSNMP\_GetBulk

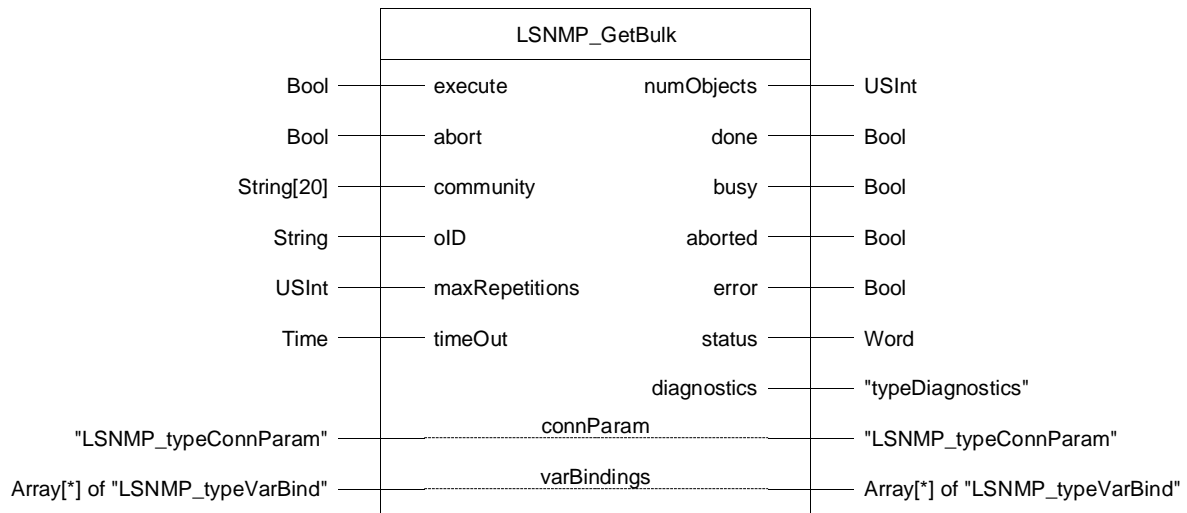


Table 8-4: Parameters of LSNMP\_GetBulk

Name	Declaration	Data type	Comment
execute	Input	Bool	The job is started with a positive edge. "abort" must be FALSE.
abort	Input	Bool	With a positive edge, the current job is canceled. The command is only accepted if the block is currently busy.
community	Input	String[20]	Community
oid	Input	String	Object Identifier of the first requested tag according to dot notation, e.g. '1.3.6.1.2.1.1.5.0'
maxRepetitions	Input	USInt	Max. number of objects to be returned
timeOut	Input	Time	Time after which a job should be automatically canceled
numObjects	Output	USInt	Number of objects received
done	Output	Bool	TRUE: Job successfully executed. The received data is available at the "varBindings" parameter.
busy	Output	Bool	TRUE: Job is being executed
aborted	Output	Bool	TRUE: Job was canceled
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section <a href="#">8.8</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
connParam	InOut	<a href="#">"LSNMP_typeConnParam"</a>	Connection parameters
varBindings	InOut	Array[*] of <a href="#">"LSNMP_typeVarBind"</a>	Received variable bindings

## 8.4 LSNMP\_Set

### Description

The "LSNMP\_Set" block sends a write job to an SNMP agent via the SNMP SetRequest command for an SNMP tag.

In the SetResponse telegram the block receives the result of the write request from the SNMP agent. If the tag read in does not match the written value, an error will be read out. The Write job must be set again.

### Parameter

Figure 8-4: LSNMP\_Set

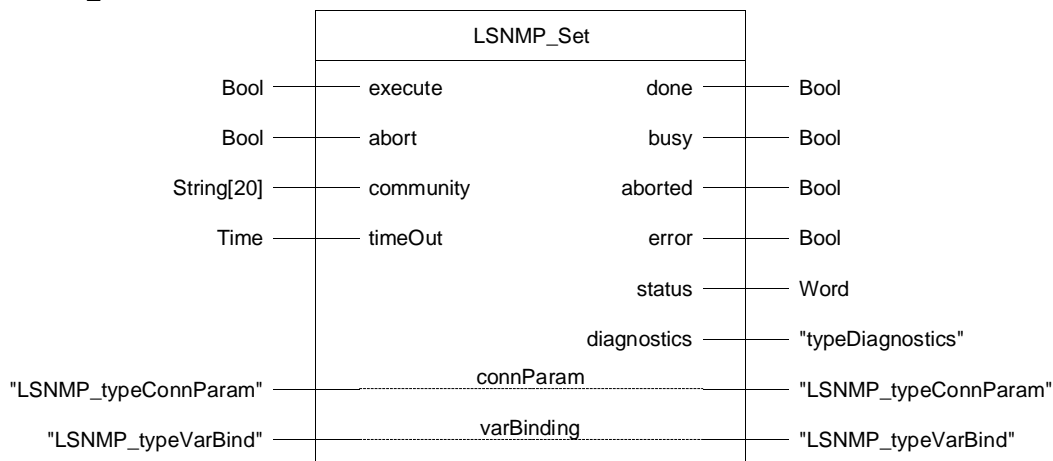


Table 8-5: Parameters of LSNMP\_Set

Name	Declaration	Data type	Comment
execute	Input	Bool	The job is started with a positive edge. "abort" must be FALSE.
abort	Input	Bool	With a positive edge, the current job is canceled. The command is only accepted if the block is currently busy.
community	Input	String[20]	Community
timeOut	Input	Time	Time after which a job should be automatically canceled
done	Output	Bool	TRUE: Job successfully executed
busy	Output	Bool	TRUE: Job is being executed
aborted	Output	Bool	TRUE: Job was canceled
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section <a href="#">8.8</a> )
diagnostics	Output	"typeDiagnostics"	Advanced diagnostic information (see Section <a href="#">12.2</a> )
connParam	InOut	"LSNMP_typeConnParam"	Connection parameters
varBinding	InOut	"LSNMP_typeVarBind"	Tag binding to be sent

## 8.5 LSNMP\_SendTrap

### Description

SNMP agents (e.g. an S7 CPU) send messages to the network management system via SNMP traps to display changes in the network status.

The "LSNMP\_SendTrap" block is a parameterizable function block for sending SNMP traps.

#### Note

The SNMP agents do not receive acknowledgement for sent traps.

### Parameter

Figure 8-5: LSNMP\_SendTrap

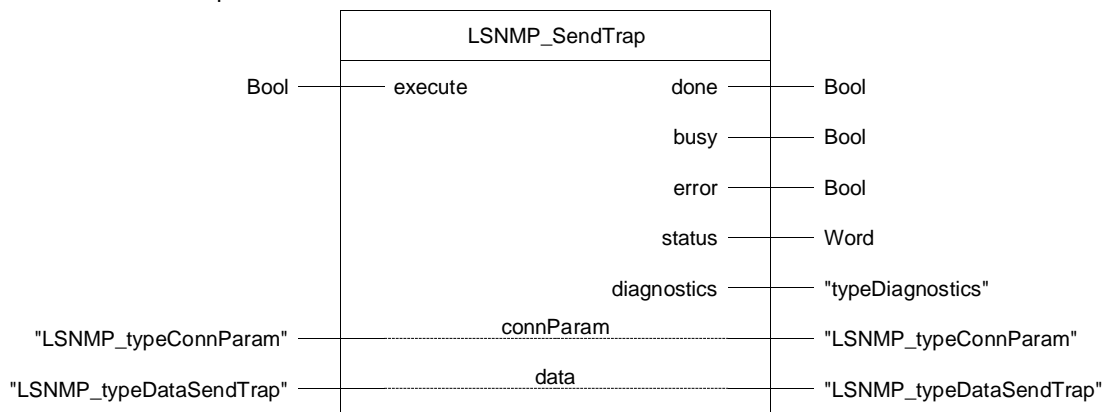


Table 8-6: Parameters of LSNMP\_SendTrap

Name	Declaration	Data type	Comment
execute	Input	Bool	The job is started with a positive edge.
done	Output	Bool	TRUE: Job successfully executed
busy	Output	Bool	TRUE: Job is being executed
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section 8.8)
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section 12.2)
connParam	InOut	<a href="#">"LSNMP_typeConnParam"</a>	Connection parameters
data	InOut	<a href="#">"LSNMP_typeDataSendTrap"</a>	Data to be sent with the SNMP trap



## 8.6 PLC Data Types

### LSNMP\_typeConnParam

The PLC data type "LSNMP\_typeConnParam" contains parameters that are required to establish a connection to the partner.

Table 8-7: Parameters of LSNMP\_typeConnParam

Name	Data type	Comment
hwIdentifier	HW_ANY	Hardware identification of the Ethernet interface
connID	Word	Connection ID
ipAddress	IP_V4	IP address of the partner
localPort	UInt	Local port

**Note**

If you want to run multiple blocks of the library or instances of the same block at the same time, the parameters "connID" and "localPort" must be unique for each instance.

### LSNMP\_typeDataSendTrap

The PLC data type "LSNMP\_typeDataSendTrap" contains all parameterizable data that can be sent to an SNMP manager with "LSNMP\_SendTrap".

Table 8-8: Parameters of LSNMP\_typeDataSendTrap

Name	Data type	Comment
community	String[20]	Community
oID	String	Object Identifier of the agent generating the trap, in dot notation, e.g. '1.3.6.1.2.1.1.5.0'
ipAgent	IP_V4	IP address of the agent generating the trap
genericTrap	Int	General Trap ID <ul style="list-style-type: none"> <li>0: Cold start (coldStart)</li> <li>1: Warm start (warmStart)</li> <li>2: Link Down (linkDown)</li> <li>3: Link Up (linkUp)</li> <li>4: Authentication failure (authenticationFailure)</li> <li>5: EGP neighbor lost (egpNeighborLoss)</li> <li>6: company-specific (enterpriseSpecific)</li> </ul>
specificTrap	Int	Company specific trap ID If the general trap ID "6" is selected, a company-specific trap ID can be sent with this parameter. Otherwise, the company-specific trap ID is "0".
varBinding	"LSNMP_typeVarBind"	Optional tag binding to be sent with the trap. In this case, the trap ID is considered sufficient information.

**LSNMP\_typeVarBind**

With SNMP, tags are transmitted as "tag bindings". A tag binding consists of the Object Identifier of the tag and the actual value. The PLC data type "LSNMP\_typeVarBind" defines such a tag binding. So that the block or the user can interpret the value correctly, the data type and the length are also specified.

Table 8-9: Parameters of LSNMP\_typeVarBind

Name	Data type	Comment
oid	String	Object Identifier of the tag in dot notation, e.g. '1.3.6.1.2.1.1.5.0'
type	Byte	Data type of the tag Common data types: 16#02: Integer 16#04: String 16#41: Counter 16#43: Timeticks
length	Int	Tag length
value	Array[0..255] of byte	Value of the tag

**8.7 Integration into the User Project**

You will find a detailed application example for the integration of the library into your user project in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/57249109>

## 8.8 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The following are the status and error codes specific to the "LSNMP" library.

Table 8-10: Status and error codes

Code	Description
16#0000	Job completed successfully
16#0FFF	Job was canceled
16#7000	No job active
16#7001	First call of the command
16#7002	Follow-up call of the command
16#7003	Cancels the job
16#8107	The received value is too large for the memory area at the "tagBinding" or "tagBindings" parameter.
16#8109	The received packet is too large for the internal reception area.
16#8408	Time-out: The job could not be completed within the specified time. Possible reasons for a time-out: <ul style="list-style-type: none"> <li>• Partner is not available</li> <li>• Community is wrong</li> </ul>
16#8501	SNMP error: Reply telegram is too large
16#8502	SNMP error: Invalid OID
16#8503	SNMP error: Invalid value
16#8504	SNMP error: No write access
16#8505	SNMP error: General error
16#8600	The FB is in an invalid state.
16#8601	Error in subordinate command "TCON" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8602	Error in subordinate command "TUSEND" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8603	Error in subordinate command "TURCV" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.

# 9 LSNTP

## 9.1 Overview

### 9.1.1 Range of Functions

For automation cells or subsystems, it is often secondary to use the exact "atomic time". It is usually sufficient to have a common time base for all automation components.

The use of a SIMATIC S7 CPU as an SNTP server enables flexible and simple synchronization of systems and subsystems, for example, to obtain meaningful timestamps for error messages and logging data system-wide.

The library provides a function block that performs the following functions:

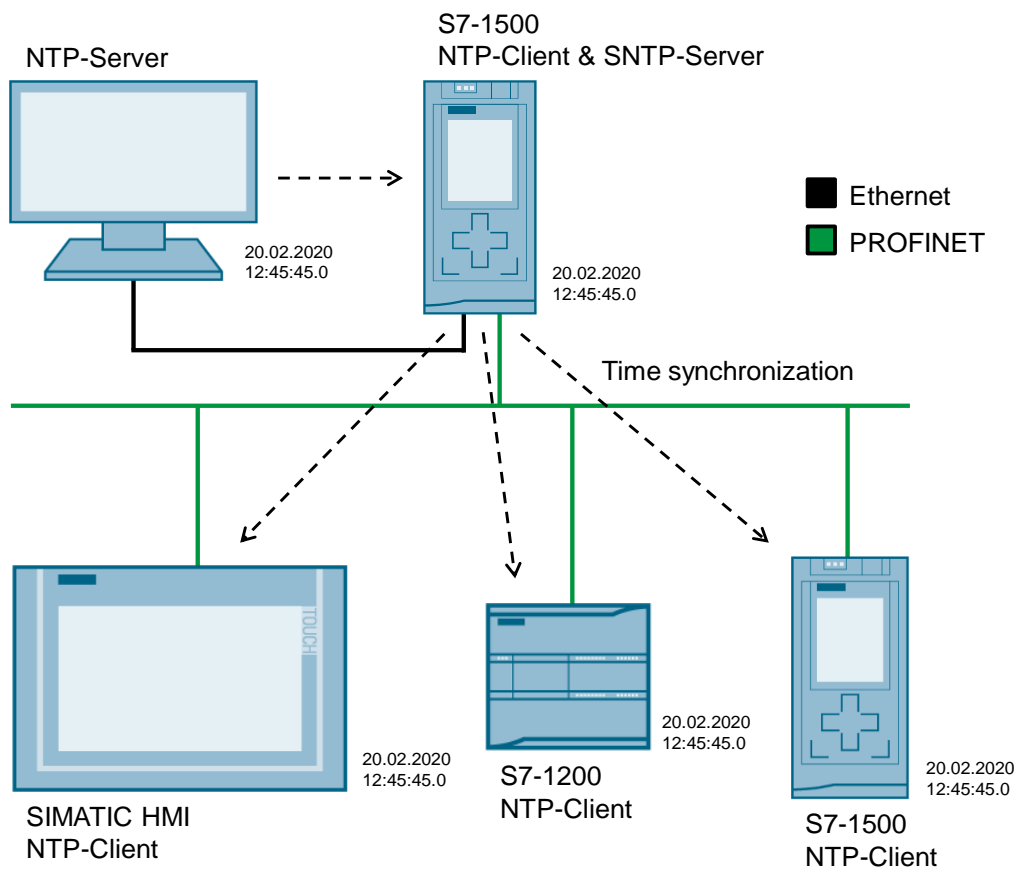
- Receiving and evaluating an NTP telegram from an (S)NTP Client
- Creating and sending an NTP telegram to the client for time synchronization

The inaccuracy of the SNTP server is less than 10 ms.

### Example scenario

The following figure shows a possible example configuration with a SIMATIC S7-1500 CPU as the SNTP server. Here, the CPU synchronizes its time with an external NTP server. However, configurations with other clocks are also possible.

Figure 9-1: Example scenario



## 9.1.2 Components of the Library

The "LSNTP" library consists of the following blocks and data types.

### Function blocks

Table 9-1: Function blocks

Name	Type	Version	Description
LSNTP_Server	FB	V4.0.0	Realizes the function of the SNTTP server.

### PLC Data Types

Table 9-2: PLC data types

Name	Type	Version	Description
LSNTP_typeTelegram	Data type	V1.0.0	Describes the structure of the NTP telegram.
LSNTP_typeTimestamp	Data type	V1.0.0	Describes the structure of the timestamp.

## 9.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.2
- SIMATIC ET 200SP Open Controllers as of firmware V2.0
- SIMATIC S7-1500 Software Controllers as of Firmware V2.0
- CP 1543-1, CP 1542SP-1, CP 1542SP-1 IRC, CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V2.0

## 9.2 LSNTP\_Server

### 9.2.1 Interface Description

#### Description

The function block implements the function of the SNTP server.

#### Parameter

Figure 9-2: LSNTP\_Server

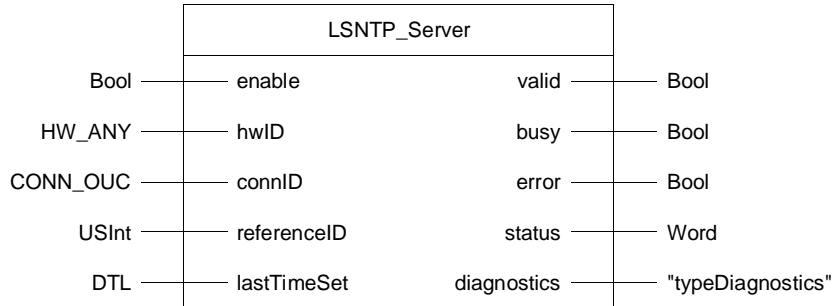


Table 9-3: Parameter of LSNTP\_Server

Name	Declaration	Data type	Comment
enable	Input	Bool	TRUE: Activates the SNTP server
hwID	Input	HW_ANY	Hardware identifier of the PN/IE interface
connID	Input	CONN_OUC	Unique connection ID
referenceID	Input	USInt	The input specifies from which time source the server CPU obtains the time: <ul style="list-style-type: none"> <li>0: uncalibrated (set "by hand")</li> <li>1: primary reference (e.g., DCF 77)</li> <li>2: secondary reference (e.g., from GPS receiver)</li> </ul> The information is passed on to the NTP client in the SNTP protocol.
lastTimeSet	Input	DTL	Time specification when the time of the controller was last set. This information is passed on to the NTP client in the SNTP protocol. If the parameter is not assigned, the current time of the CPU is transferred instead.
valid	Output	Bool	TRUE: The block is working without error
busy	Output	Bool	TRUE: The block is processed
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section <a href="#">9.2.2</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )

## 9.2.2 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The status and error codes specific to the "LSNTP" library are listed below.

Table 9-4: Status and error codes

Code	Description
16#0000	NTP telegram sent to an (S)NTP client
16#0001	NTP telegram of an (S)NTP client received
16#7000	The block is not executed
16#7001	First call of the command
16#7002	Follow-up call of the command
16#7003	Block is deactivated
16#7F01	Invalid package received
16#8600	The FB is in an invalid state.
16#8601	Error in subordinate command "TCON" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8602	Error in subordinate command "TUSEND" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8603	Error in subordinate command "TURCV" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.

# 10 LSyslog

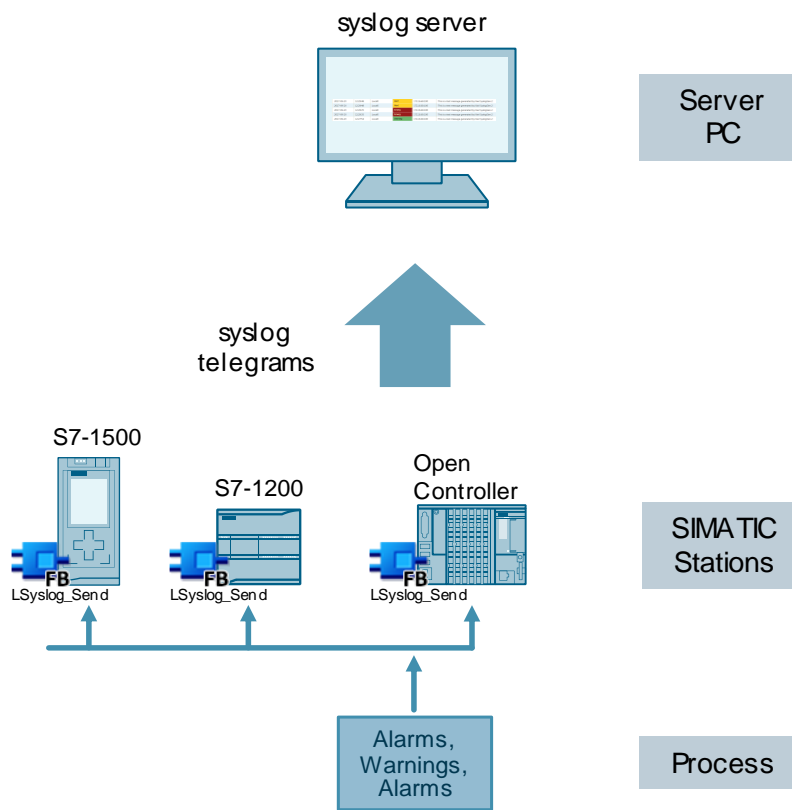
## 10.1 Overview

### 10.1.1 Range of Functions

With the simply designed syslog protocol, applications can send messages, warnings, or error conditions to a syslog server. syslog is typically used for computer system management and security monitoring and has now established itself as a standard (RFC 5424) in logging.

This library emulates the syslog protocol and offers the possibility to send messages via UDP or TCP with optional TLS encryption to a syslog server, such as SINEC INS.

Figure 10-1





## 10.1.2 Components of the Library

The "LSyslog" library contains the following objects.

### Function blocks

Table 10-1: Function blocks

Name	Version	Description
LSyslog_Send	V2.0.0	Organizes the transmission and creation of the telegrams.

### PLC Data Types

Table 10-2: PLC data types

Name	Version	Description
LSyslog_typeMessage	V1.1.0	This data type describes all tag data of a syslog message.

## 10.1.3 Validity

The library is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.4
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- CP 1543-1, CP 1542SP-1, CP 1542SP-1 IRC, CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V2.0

## 10.2 LSyslog\_Send

### 10.2.1 Interface Description

#### Description

The FB "LSyslog\_Send" establishes a connection to the syslog server via UDP or TCP with optional TLS encryption and sends syslog messages to the syslog server upon request. Afterwards, the connection is automatically broken again.

#### Parameter

Figure 10-2: LSyslog\_Send

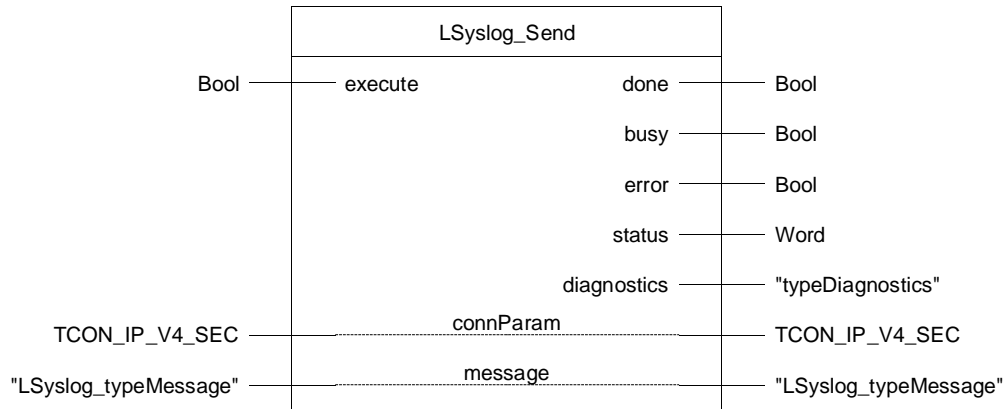


Table 10-3: Parameters of LSyslog\_Send

Name	Declaration	Data type	Comment
execute	Input	Bool	With a positive edge, the connection to the syslog server is established and the syslog message is sent.
done	Output	Bool	TRUE: The send job was successfully executed
busy	Output	Bool	TRUE: The block is executed
error	Output	Bool	TRUE: An error has occurred
status	Output	Word	Status and error codes (see Section <a href="#">10.2.2</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
connParam	InOut	TCON_IP_V4_SEC	Connection parameters
message	InOut	<a href="#">"LSyslog_typeMessage"</a>	syslog message to be sent

In the FB the octet stuffing method for syslog is implemented over TCP by appending a line feed to the end of the message.

**Note**

The octet counting method is also implemented, but commented out.

If you want to use the octet counting method for your application instead, you can adjust the comments in the build message region accordingly.

**Note**

You will find an application example for the usage of certificates in TIA Portal in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/109769068>

**10.2.2 Error Handling**

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The following are the status and error codes specific to the "LSyslog" library.

Table 10-4: Status and error codes

Code	Description
16#0000	Job completed successfully
16#7000	No job active
16#7001	First call of the command
16#7002	Follow-up call of the command
16#8408	Time-out: The job could not be completed within the specified time. Possible reasons for a time-out: <ul style="list-style-type: none"> <li>• Partner is not available</li> </ul>
16#8600	The FB is in an invalid state.
16#8601	Error in subordinate command "TCON" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8602	Error in subordinate command "TUSEND" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.
16#8603	Error in subordinate command "TSEND" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system.

## 10.3 PLC Data Types

### LSyslog\_typeMessage

The PLC data type describes all parameters of the syslog message to be sent.

Table 10-5: Parameters of LSyslog\_typeMessage

Name	Type	Comment
facility	Int	Facility (origin of the message) <ul style="list-style-type: none"> <li>• 0: kernel messages</li> <li>• 1: user-level messages</li> <li>• 2: mail system</li> <li>• 3: system daemons</li> <li>• 4: security/authorization messages</li> <li>• 5: messages generated internally by syslogd</li> <li>• 6: line printer subsystem</li> <li>• 7: network news subsystem</li> <li>• 8: UUCP subsystem</li> <li>• 9: clock daemon</li> <li>• 10: security/authorization messages</li> <li>• 11: FTP daemon</li> <li>• 12: NTP subsystem</li> <li>• 13: log audit</li> <li>• 14: log alert</li> <li>• 15: clock daemon (note 2)</li> <li>• 16-23: local0-local7</li> </ul>
severity	Int	Severity (severity of the message) <ul style="list-style-type: none"> <li>• 0: Emergency</li> <li>• 1: Alert</li> <li>• 2: Critical</li> <li>• 3: Error</li> <li>• 4: Warning</li> <li>• 5: Notice</li> <li>• 6: Informational</li> <li>• 7: Debug</li> </ul>
hostname	String	Host name that identifies the machine that originally sent the syslog message, e.g. FQDN, IP address, or PLC name. If no host name is used, the tag must be set to "-".
appName	String	Application name that identifies the device or application that generated the message. If no application name is used, the tag must be set to "-".
msgID	String	Message ID that identifies the type of message. If no message ID is used, the tag must be set to "-".
message	String	Message text in free form

## 10.4 Integration into the User Project

You will find a detailed application example for the integration of the library into your user project in the Siemens Industry Online Support:

<https://support.industry.siemens.com/cs/ww/en/view/51929235>

# 11 Master Copies

## 11.1 OUC Master Copies

### 11.1.1 Overview

#### 11.1.1.1 Range of Functions

TCP/IP-based Open User Communication (OUC) has become the standard for communication with SIMATIC Controllers.

In SIMATIC Controller OUC is implemented on the basis of commands (e.g. TCON, TSEND, TRCV, and TDISCON). The user parameterizes the commands in their user program and calls them in an error-tolerant manner.

To facilitate this, we offer function blocks in SCL as master copies.

The FBs call the OUC commands in the order and manner recommended in the manuals. In addition, FBs already contain the following mechanisms:

- Connection management with the commands "TCON" and "TDISCON"
- Send data to a partner
- Receive data from a partner

You can use the FBs as a template for your own communication projects or the implementation of other IP-based protocols.

#### 11.1.1.2 Components

The following FBs are available as master copies:

Table 11-1: Master copies

Name	Description
IsoOnTcpTemplate	Master copy to implement communication via ISO-on-TCP protocol
TcpTemplate	Master copy to implement communication via TCP protocol
TcpSecTemplate	Master copy to implement secured communication via TCP protocol
UdpTemplate	Master copy to implement communication via UDP protocol

#### 11.1.1.3 Validity

The master copies "IsoOnTcpTemplate", "TcpTemplate" and "UdpTemplate" are valid for:

- SIMATIC S7-1500 Controller as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.2
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- SIMATIC S7-300 Controllers
- SIMATIC S7-400 Controllers
- CP 1543-1, CP 1542SP-1, CP 1542SP-1 IRC, CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V2.0

The master copy "TcpSecTemplate" is valid for:

- SIMATIC S7-1500 Controllers as of firmware V2.0
- SIMATIC S7-1200 Controllers as of firmware V4.4
- SIMATIC ET 200SP Open Controllers as of firmware V2.5
- SIMATIC S7-1500 Software Controllers as of firmware V2.5
- CP 1543-1 as of firmware V2.0
- CP 1543SP-1 as of firmware V1.0
- CP 1243-1, CP 1243-8 as of firmware V3.2

### 11.1.2 IsoOnTcpTemplate

#### Functional description

The FB "IsoOnTcpTemplate" implements a complete ISO-on-TCP communication relation to a partner. It encapsulates all OUC commands in a user-friendly shell to perform the following functions:

- Connection establishment and termination via the input "enable"
- Send user data of length "sendLen" to the partner via the "sendData" input as soon as the "send" input detects a positive edge.
- Receive data from a partner and output it in the receive area at the "rcvdData" parameter.
- Output the status of transmission and connection at the output parameter "status".

**Note**

If the received data is stored in an optimized data block, the input parameter "rcvLen" must contain the value 0. For a standard data block, this input parameter must contain the number of bytes to be received.

#### Parameter

Figure 11-1

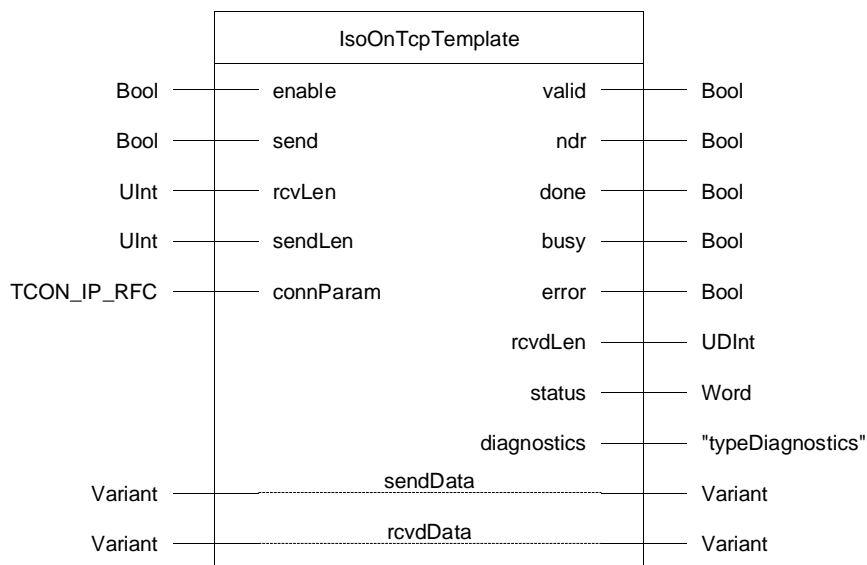


Table 11-2

Name	Declaration	Data type	Comment
enable	Input	Bool	Release signal for connection setup and data exchange
send	Input	Bool	Triggering the send job
rcvLen	Input	UInt	Receive data length If the receive data is stored in an optimized DB, then rcvLen = 0.
sendLen	Input	UInt	Maximum number of bytes sent with the job
connParam	Input	TCON_IP_RFC	Connection parameters
valid	Output	Bool	TRUE: The block executes its function without error
ndr	Output	Bool	TRUE: New data has been received
done	Output	Bool	TRUE: The send job was successfully executed
busy	Output	Bool	TRUE: The block is executed
error	Output	Bool	TRUE: An error has occurred
rcvdLen	Output	UDInt	Length of received data in bytes
status	Output	Word	Status and error codes (see Section <a href="#">11.1.6</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
sendData	InOut	Variant	Transmit data range
rcvdData	InOut	Variant	Receive data range

### 11.1.3 TcpTemplate

#### Functional description

The FB "TcpTemplate" implements a complete TCP communication relationship to a partner. It encapsulates all OUC commands in a user-friendly shell to perform the following functions:

- Connection establishment and termination via the input "enable"
- Send user data of length "sendLen" to the partner via the "sendData" input as soon as the "send" input detects a positive edge.
- Receive data from a partner and output it in the receive area at the "rcvdData" parameter.
- Output the status of transmission and connection at the output parameter "status".

**Note**

Enable the Adhoc mode to receive telegrams with dynamic data length. In this case the input parameter "rcvLen" is irrelevant.

Deactivate the Adhoc mode to receive telegrams with fixed data length. In this case you have to specify the number of bytes to be received at the input parameter "rcvLen".

#### Parameter

Figure 11-2

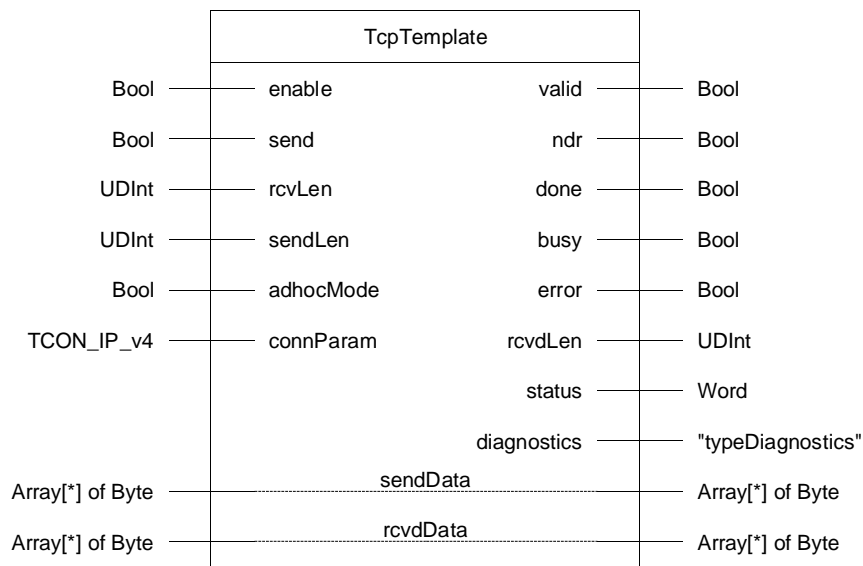


Table 11-3

Name	Declaration	Data type	Comment
enable	Input	Bool	Release signal for connection setup and data exchange
send	Input	Bool	Triggering the send job



Name	Declaration	Data type	Comment
rcvLen	Input	UDInt	Receive data length <ul style="list-style-type: none"> <li>S7-1500 CPUs: maximum 65536 bytes</li> <li>S7-1200 CPUs maximum 8192 bytes</li> </ul> <b>Note</b> Parameter is irrelevant when Adhoc mode is enabled. As much data is read as is currently available. The max. data length is defined by the length of the receive area referenced by rcvData.
sendLen	Input	UDInt	Maximum number of bytes sent with the job. <ul style="list-style-type: none"> <li>S7-1500 CPUs: maximum 65536 bytes</li> <li>S7-1200 CPUs maximum 8192 bytes</li> </ul>
adhocMode	Input	Bool	TRUE (Adhoc enabled): <ul style="list-style-type: none"> <li>The data is available immediately.</li> <li>Data reception with dynamic data length</li> </ul> FALSE (Adhoc disabled): <ul style="list-style-type: none"> <li>The data is available as soon as the data length specified at parameter LEN has been received completely.</li> <li>Data reception with specified data length.</li> </ul>
connParam	Input	TCON_IP_v4	Connection parameters
valid	Output	Bool	TRUE: The block executes its function without error
ndr	Output	Bool	TRUE: New data has been received
done	Output	Bool	TRUE: The send job was successfully executed
busy	Output	Bool	TRUE: The block is executed
error	Output	Bool	TRUE: An error has occurred
rcvdLen	Output	UDInt	Length of received data in bytes
status	Output	Word	Status and error codes (see Section <a href="#">11.1.6</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information
sendData	InOut	Array[*] of bytes	Transmit data range
rcvdData	InOut	Array[*] of bytes	Receive data range

### 11.1.4 TcpSecTemplate

#### Functional description

The FB "TcpSecTemplate" implements a complete TCP communication relationship to a partner.

With the FB "TcpSecTemplate" and with S7-1200 CPUs as of firmware V4.4 and S7-1500 CPUs from firmware V2.0, it is possible to parameterize the communication connections at TCP via IPv4 by means of Secure Communication.

It encapsulates all OUC commands in a user-friendly shell to perform the following functions:

- Connection establishment and termination via the input "enable"
- Send user data of length "sendLen" to the partner via the "sendData" input as soon as the "send" input detects a positive edge
- Receive data from a partner and output it in the receive area at the "rcvdData" parameter.
- Output the status of transmission and connection at the output parameter "status"

**Note**

Enable Adhoc mode to receive telegrams with dynamic data length. In this case the input parameter "rcvLen" is irrelevant.

Deactivate the Adhoc mode to receive telegrams with fixed data length. In this case you have to specify the number of bytes to be received at the input parameter "rcvLen".

#### Parameter

Figure 11-3

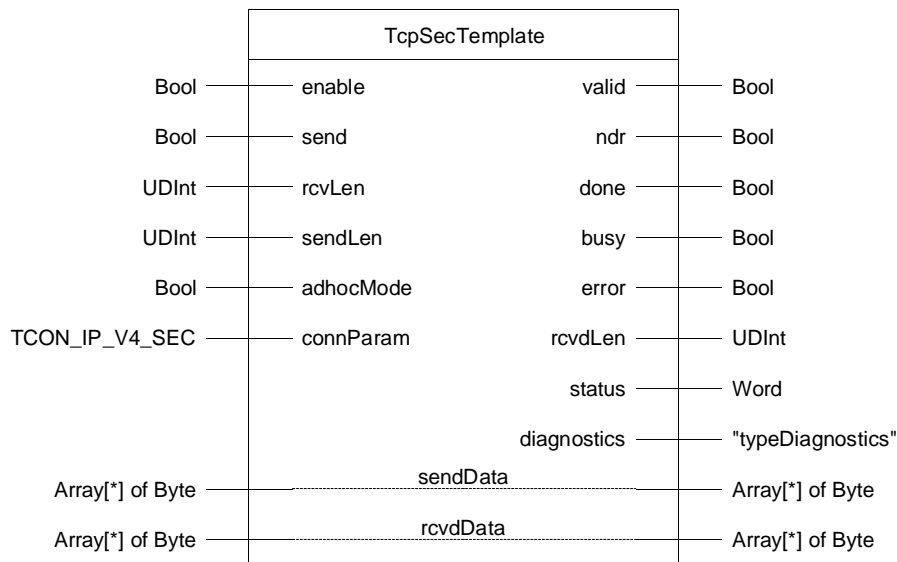


Table 11-4

Name	Declaration	Data type	Comment
enable	Input	Bool	Release signal for connection setup and data exchange
send	Input	Bool	Triggering the send job
rcvLen	Input	UDInt	Receive data length <ul style="list-style-type: none"> <li>S7-1500 CPUs: maximum 65536 bytes</li> <li>S7-1200 CPUs maximum 8192 bytes</li> </ul> <b>Note</b> Parameter is irrelevant when Adhoc mode is enabled. As much data is read as is currently available. The max. data length is defined by the length of the receive area referenced by rcvData.
sendLen	Input	UDInt	Number of bytes sent with the job. <ul style="list-style-type: none"> <li>S7-1500 CPUs: maximum 65536 bytes</li> <li>S7-1200 CPUs maximum 8192 bytes</li> </ul> <b>Note</b> If the Adhoc mode is activated, the total length of the telegram is transmitted in the first 4 bytes. These additional 4 bytes must be taken into account at the "sendLen" parameter, i.e. sendLen = 4 byte + user data
adhocMode	Input	Bool	TRUE (Adhoc enabled): <ul style="list-style-type: none"> <li>The data is available immediately.</li> <li>Data reception with dynamic data length</li> </ul> FALSE (Adhoc disabled): <ul style="list-style-type: none"> <li>The data is available as soon as the data length specified at parameter LEN has been received completely.</li> <li>Data reception with specified data length.</li> </ul>
connParam	Input	TCON_IP_V4_SEC	Connection parameters (see TIA Portal information system)
valid	Output	Bool	TRUE: The block executes its function without error
ndr	Output	Bool	TRUE: New data has been received
done	Output	Bool	TRUE: The send job was successfully executed
busy	Output	Bool	TRUE: The block is executed
error	Output	Bool	TRUE: An error has occurred
rcvdLen	Output	UDInt	Length of received data in bytes
status	Output	Word	Status and error codes (see Section <a href="#">11.1.6</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
sendData	InOut	Array[*] of bytes	Transmit data range
rcvdData	InOut	Array[*] of bytes	Receive data range

### 11.1.5 UdpTemplate

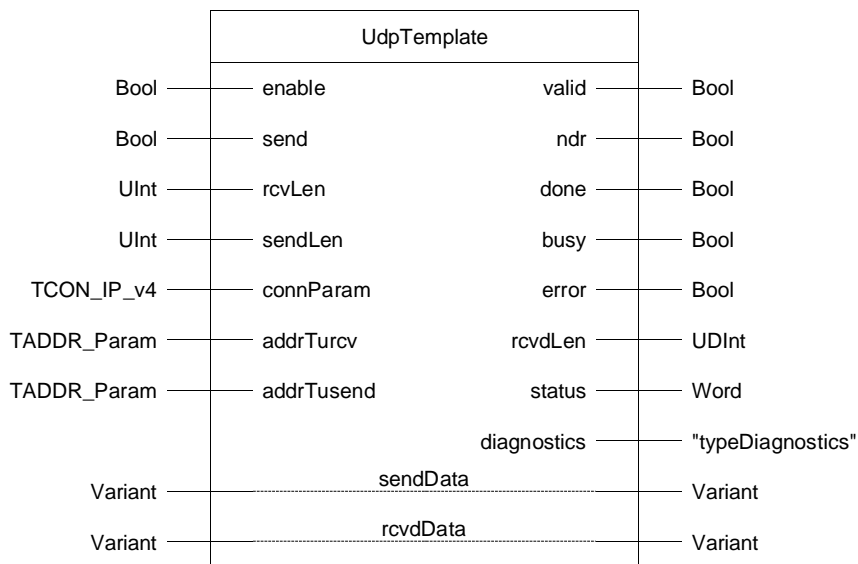
#### Functional description

The FB "UdpTemplate" implements a complete UDP communication relation to a partner. It encapsulates all OUC commands in a user-friendly shell module to perform the following functions:

- Connection establishment and termination via the input "enable"
- Send user data at the "sendData" input with the length "sendLen" to the partner as soon as the "send" input detects a positive edge
- Receive data from a partner and output it in the receive area at the "rcvdData" parameter
- Output the status of transmission and connection at the output parameter "status"

#### Parameter

Figure 11-4



© Siemens AG 2023. All rights reserved.

Table 11-5

Name	Declaration	Data type	Comment
enable	Input	Bool	Release signal for connection setup and data exchange
send	Input	Bool	Triggering the send job
rcvLen	Input	UInt	Length of the receive area in bytes: Value range: 0 (recommended) or 1 to 1472 (as of firmware version V2.5 of the S7-1500 CPUs with Unicast or Multicast: 1 to 2048)
sendLen	Input	UInt	Number of bytes that are to be sent with the job Value range: 1 to 1472 (as of firmware version V2.5 of the S7-1500 CPUs with Unicast or Multicast: 1 to 2048)

Name	Declaration	Data type	Comment
connParam	Input	TCON_IP_v4	Connection parameters Detailed information can be found in the TIA Portal information system.
addrTurcv	Input	TADDR_Param	Address information of the communication partner for the "TURCV" command Detailed information can be found in the TIA Portal information system.
addrTusend	Input	TADDR_Param	Address information of the communication partner for the "TSEND" command Detailed information can be found in the TIA Portal information system.
valid	Output	Bool	TRUE: The block executes its function without error
ndr	Output	Bool	TRUE: New data has been received
done	Output	Bool	TRUE: The send job was successfully executed
busy	Output	Bool	TRUE: The block is executed
error	Output	Bool	TRUE: An error has occurred
rcvdLen	Output	UDInt	Length of received data in bytes
status	Output	Word	Status and error codes (see Section <a href="#">11.1.6</a> )
diagnostics	Output	<a href="#">"typeDiagnostics"</a>	Advanced diagnostic information (see Section <a href="#">12.2</a> )
sendData	InOut	Variant	Transmission range, contains address and length The address refers to: <ul style="list-style-type: none"> <li>• A data block</li> <li>• A marker</li> <li>• The process image of inputs</li> <li>• The process image of outputs</li> </ul> When structures are transmitted, the structures on the transmit and receive sides must be identical.
rcvdData	InOut	Variant	Receive area The address refers to: <ul style="list-style-type: none"> <li>• A data block</li> <li>• A marker</li> <li>• The process image of inputs</li> <li>• The process image of outputs</li> </ul> When structures are transmitted, the structures on the transmit and receive sides must be identical.

**Note**

When sending more than 1472 bytes, you must ensure that your receiver supports more than 1472 bytes. If this is not the case, you will not notice the failed reception on the transmitter side.

### 11.1.6 Error Handling

General information about the status outputs and the diagnostics of the blocks of these libraries can be found in Section [12.2](#).

The following are the status and error codes specific to the OUC master copies.

Table 11-6: Status and error codes

Code	Meaning
16#0000	Job completed successfully
16#7000	No job active
16#7001	First call of the command
16#7002	Follow-up call of the command
16#8408	Time-out: The job could not be completed within the specified time. Possible reasons for a time-out: <ul style="list-style-type: none"> <li>• Partner is not available</li> </ul>
16#8600	The FB is in an invalid state.
16#8601	Error in subordinate command "TCON" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system or the STEP 7 online help.
16#8602	Error in subordinate command "TSEND" or "TUSEND" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system or the STEP 7 online help.
16#8603	Error in subordinate command "TRCV" or "TURCV" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system or the STEP 7 online help.
16#8604	Error in subordinate command "TDISCON" The error code of the command is output to "diagnostics.subfunctionStatus". For the meaning of the respective error code, refer to the TIA Portal information system or the STEP 7 online help.

## 11.2 OPC UA structured data types

Under "Master Copies" you will find the folder "OPC UA data types", which provides you with pre-defined OPC UA structure data types as UDTs. This selection of structured data types is usually used as OPC UA properties and can be mapped using "SiOME" ([\10](#)), for example.

## 12 Useful Information

### 12.1 Libraries in TIA Portal

Most of the blocks are stored as types in the library. Thus, the blocks are versioned and can use the following advantages:

- Central update function for library elements
- Versioning of library elements

**Note**

For information on library use in general, see the Guide to Library Use:

<https://support.industry.siemens.com/cs/ww/en/view/109747503>

**Note**

All blocks in the library were created in accordance with the programming style guide:

<https://support.industry.siemens.com/cs/ww/en/view/81318674>

For more information on libraries in the TIA Portal:

- How do you open, edit and upgrade global libraries in the TIA Portal?  
<https://support.industry.siemens.com/cs/ww/en/view/37364723>
- TIA Portal in under 10 minutes: Time Savers – Global libraries  
<https://support.industry.siemens.com/cs/ww/en/view/78529894>
- Which elements from STEP 7 (TIA Portal) and WinCC (TIA Portal) can be stored in a library as a type or as a master copy?  
<https://support.industry.siemens.com/cs/ww/en/view/109476862>
- When starting the TIA Portal V13 and higher, how do you get a global library to open automatically and use it as corporate library, for example?  
<https://support.industry.siemens.com/cs/ww/en/view/100451450>

## 12.2 Diagnostics

The blocks use a uniform diagnostics concept according to the programming style guide for SIMATIC S7-1200/1500.

All blocks have the outputs "busy", "error", "status", and "diagnostics". Depending on the type of block (continuous or one-time asynchronous processing), it can also have an output of "valid" or "done".

If the block implements both continuous and one-time asynchronous functions (e.g. connection setup and monitoring and sending a message), it can also have both outputs.

The "valid" or "done", "busy", "error", and "status" outputs show the current status and errors, while the "diagnostics" output provides a diagnostic structure with detailed information in the event of an error.

### Reset behavior of the outputs for blocks with "execute"

Note

As long as the input "execute" is set, the outputs keep their value. If the "execute" input is reset before the processing of the FB is completed, the values of the output parameters are output for one cycle after the job is processed.

For more information, refer to the programming style guide for SIMATIC S7-1200/1500:

<https://support.industry.siemens.com/cs/ww/en/view/81318674>

### status

The "status" output provides current information as well as errors according to the following value range.

Detailed information on the status/error codes can be found in the "Error Handling" Section of the respective library.

Table 12-1: Value ranges for information

Information	Value range
Job completed, no warning and no further details	16#0000
Job completed, further detailing	16#0001..16#0FFF
No job in process (also initial value)	16#7000
First call after input of a new job (rising edge on "execute")	16#7001
Subsequent call during active processing without further details	16#7002
Follow-up call during active machining with further detail. Occurred warnings that do not further affect the operation.	16#7003..16#7FFF



Table 12-2: Value ranges for errors

Errors	Value range
Incorrect operation of the function block	16#8001..16#81FF
Error in the parameterization	16#8200..16#83FF
Faults when executing from outside (e.g. wrong I/O signals, axis not referenced)	16#8400..16#85FF
Faults when executing internally (e.g. when a system function is called)	16#8600..16#87FF
Reserved	16#8800..16#8FFF
User-defined error classes	16#9000..16#FFFF

**diagnostics**

In the event of an error, the "diagnostics" output gives detailed information about the pending error.

The values of the "diagnostics" output are only written when an error occurs and are retained until a new job is started or another error has occurred and overwrites the values.

Unless otherwise described, the "diagnostics" output uses the PLC data type "typeDiagnostics", which is structured as follows:

Table 12-3: PLC data type typeDiagnostics

Name	Data type	Description
status	Word	Status of the FB at the time of the error.
subfunctionStatus	DWord	Status or returned value from internal commands or FBs where an error occurred. For detailed information, refer to the online help for the respective command or the documentation of the respective FB.
stateNumber	DInt	State of the internal state machine in which the error occurred.

# 13 Appendix

## 13.1 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](https://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](https://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](https://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](https://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](https://support.industry.siemens.com/cs/ww/en/sc/2067)

## 13.2 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 process – directly and independently of time and location:

[mall.industry.siemens.com](https://mall.industry.siemens.com)

## 13.3 Links and Literature

Table 13-1

No.	Topic
\1\	Siemens Industry Online Support <a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a>
\2\	Link to the entry page of the library <a href="https://support.industry.siemens.com/cs/ww/en/view/109780503">https://support.industry.siemens.com/cs/ww/en/view/109780503</a>
\3\	Application example for library "LCom" as well as blocks for S7-300/400 and SIMOTION <a href="https://support.industry.siemens.com/cs/ww/en/view/48955385">https://support.industry.siemens.com/cs/ww/en/view/48955385</a>
\4\	Application example for the "LMQTT" library <a href="https://support.industry.siemens.com/cs/ww/en/view/109748872">https://support.industry.siemens.com/cs/ww/en/view/109748872</a>
\5\	Application example for the "LMindConn" library <a href="https://support.industry.siemens.com/cs/ww/en/view/109772284">https://support.industry.siemens.com/cs/ww/en/view/109772284</a>
\6\	Application example for OPC UA PubSub UDP <a href="https://support.industry.siemens.com/cs/ww/en/view/109782455">https://support.industry.siemens.com/cs/ww/en/view/109782455</a>
\7\	Application example for OPC UA PubSub MQTT <a href="https://support.industry.siemens.com/cs/ww/en/view/109797826">https://support.industry.siemens.com/cs/ww/en/view/109797826</a>
\8\	Application example for the "LSNMP" library <a href="https://support.industry.siemens.com/cs/ww/en/view/57249109">https://support.industry.siemens.com/cs/ww/en/view/57249109</a>
\9\	Application example for the "LSyslog" library <a href="https://support.industry.siemens.com/cs/ww/en/view/51929235">https://support.industry.siemens.com/cs/ww/en/view/51929235</a>
\10\	Siemens OPC UA Modelling Editor (SiOME) <a href="https://support.industry.siemens.com/cs/ww/en/view/109755133">https://support.industry.siemens.com/cs/ww/en/view/109755133</a>

## 13.4 Change Documentation

Table 13-2

Version	Date	Change
V1.6.1	04/2023	<ul style="list-style-type: none"> <li>• LFTP: Bug fixes</li> <li>• LOpcUa: Bug fix in "LOpcUa_PubUdp" and "LOpcUa_Pub_X" FCs</li> </ul>
V1.6.0	02/2023	<ul style="list-style-type: none"> <li>• LCom: <ul style="list-style-type: none"> <li>- Added QDN as connection type</li> <li>- Added REGIONS in FB "LCom_Communication"</li> </ul> </li> <li>• LHTTP: Bug fixes</li> <li>• LMQTT: Bug fixes</li> <li>• Master copies: <ul style="list-style-type: none"> <li>- Added PLC tags "LCom_Constants"</li> <li>- Added OB "LCom_Example"</li> <li>- Added DB "LCom_ExampleComData" and "LCom_ExampleComBuffers"</li> <li>- Added data type "LCom_typeComUserData"</li> <li>- Added OPC UA structured data types</li> </ul> </li> </ul>
V1.5.0	12/2022	LOpcUa: <ul style="list-style-type: none"> <li>• Added "LOpcUa_PubMqttJson"</li> <li>• Added "LOpcUa_SubMqttJson"</li> </ul>
V1.4.0	11/2022	<ul style="list-style-type: none"> <li>• LFTP: <ul style="list-style-type: none"> <li>- Fixed incompatibility with S7-1200</li> <li>- Fixed timeout bug</li> </ul> </li> <li>• LMQTT: Minor bug fix</li> <li>• LOpcUa: <ul style="list-style-type: none"> <li>- Minor bug fixes</li> <li>- Optimizations</li> </ul> </li> <li>• Deleted "LOpcUa_typeString" and replaced with "WString"</li> </ul>
V1.3.0	05/2022	Added library "LFTP"
V1.2.0	01/2022	<ul style="list-style-type: none"> <li>• LHTTP_Get: Renamed parameter "data" to "queryParams"</li> <li>• LHTTP_PostPut: Added parameter "contentType" to make this header field adjustable</li> <li>• Minor improvements to OPC UA method templates</li> </ul>
V1.1.1	09/2021	<ul style="list-style-type: none"> <li>• Minor bug fixes in "LSNMP"</li> <li>• Fixed references in library for TIA Portal V17</li> </ul>
V1.1.0	07/2021	<ul style="list-style-type: none"> <li>• Added library "LMindConn"</li> <li>• Added library "LSNTP"</li> <li>• Added FBs for PubSub via MQTT to library "LOpcUa"</li> <li>• Master copies: <ul style="list-style-type: none"> <li>- Revised OUC templates</li> <li>- Added FB "OpcUaMethodTemplateSimple"</li> <li>- Bug fixes in "OpcUaMethodTemplateAdvanced"</li> <li>- Added OPC UA status codes as global constants</li> </ul> </li> </ul>
V1.0.1	05/2021	LHTTP: <ul style="list-style-type: none"> <li>• Bug fixes for chunked encoding</li> <li>• Supported chunk size increased</li> </ul>
V1.0.0	12/2020	First version