

The background image shows a man in a light blue shirt from the side, looking at a tablet. The scene is a factory or industrial environment with various machines and equipment. Overlaid on the image are several digital graphics: a clock face, a '24/7' icon with a circular arrow, a 'NEWS' section with a person icon, a 'Home' button, and a large 'Industry Online Support' text. There are also binary code (0s and 1s) and network-like icons scattered throughout the digital overlay.

**SIEMENS**

*Ingenuity for life*

## **Sending and receiving messages with messenger services as demonstrated with "Telegram"**

SCALANCE MUM856-1, Library\_Comm\_Controller,  
HTTPS, Telegram Messenger, LHTTP

<https://support.industry.siemens.com/cs/ww/en/view/109763879>

**Siemens  
Industry  
Online  
Support**



## Legal information

### Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

### Disclaimer of liability

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

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

### Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

### Security information

Siemens provides products and solutions with Industrial Security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place. For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: <https://www.siemens.com/industrialsecurity>.

# Table of contents

<b>Legal information .....</b>	<b>2</b>
<b>1 Introduction .....</b>	<b>4</b>
1.1 Overview .....	4
1.2 Principle of operation .....	6
1.3 Components used .....	9
<b>2 Engineering .....</b>	<b>11</b>
2.1 Hardware setup .....	11
2.2 Engineering of the STEP 7 program.....	14
2.2.1 Function block "TelegramPushNotification" .....	16
2.2.2 Global data block "DataPushNotification" .....	22
2.3 Configuring the SIMATIC S7 CPU.....	24
2.3.1 Configure the Ethernet interface .....	24
2.3.2 Certificate management.....	25
2.4 Configuring the SCALANCE MUM856-1 .....	32
2.5 Downloading the STEP 7 V17 project .....	35
2.6 Operation .....	36
2.7 Troubleshooting.....	41
<b>3 Useful information.....</b>	<b>42</b>
3.1 Telegram Messenger .....	42
3.1.1 Creating a Telegram bot .....	44
3.1.2 Find the chat ID .....	47
3.2 Alternative messenger services .....	48
<b>4 Appendix.....</b>	<b>49</b>
4.1 Service and support.....	49
4.2 Industry Mall .....	50
4.3 Links and literature .....	50
4.4 Change documentation.....	50

# 1 Introduction

## 1.1 Overview

### Starting point

Digitalization makes its effects known for most people especially through changes in communication behavior. Smartphones are constant companions and with them, the use of one or more messenger services has become an everyday activity. This applies above all in private life. But these communication habits are becoming increasingly prevalent in companies as well.

To complement emails or SMS, a company can now use messenger services in service communication as well.

Modern messenger services allow not only for communication between accounts of individuals, but also between man and machine (bots). In this way, messages can be sent to or received from multiple devices.

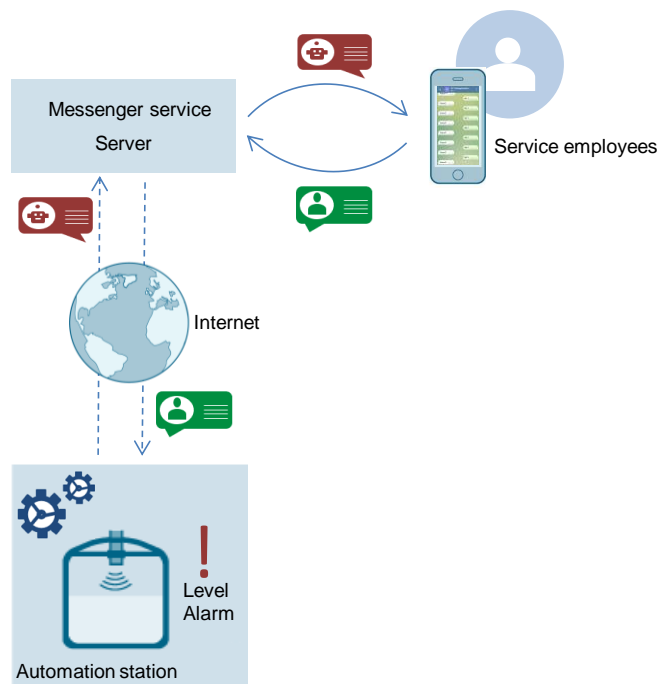
### Requirements

A SIMATIC station is to send alarm messages, warning notifications or important information about the plant status to a service worker's smartphone via a messenger service. Transmission can be cyclic or event-triggered.

In addition, the SIMATIC station is also to receive and interpret messages.

The following Figure provides an overview of the automation task.

Figure 1-1





## Solution approach

A SIMATIC S7-1500 CPU communicates over HTTPS (Hypertext Transfer Protocol Secure) with the service technician's smartphone that has the "Telegram Messenger" app installed.

The SIMATIC S7-1500 CPU is connected to the internet with the SCALANCE MUM856-1 via the cellular radio network.

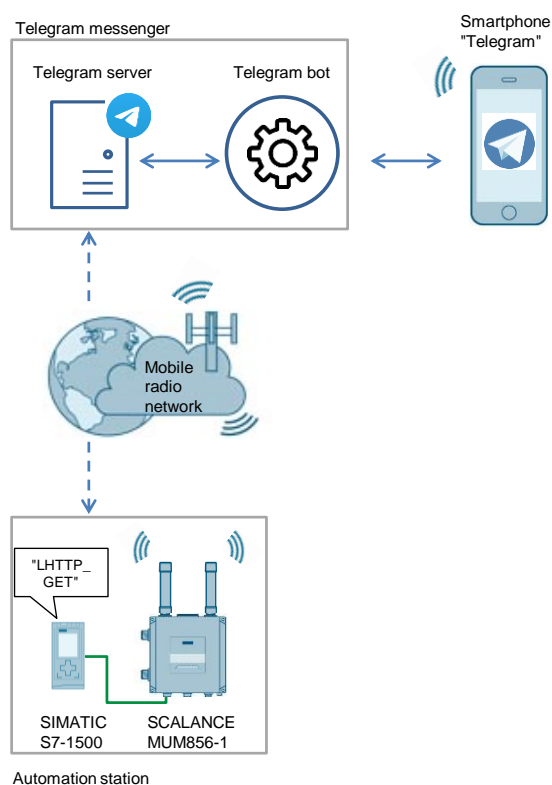
The SIMATIC S7-1500 CPU sends messages, commands and queries to the Telegram server or retrieves data from it.

The "LHTTP\_Get" block from the "LHTTP" library, part of the Libraries for Communication for SIMATIC Controllers (<https://support.industry.siemens.com/cs/ww/en/view/109780503>), is used to communicate with the Telegram server.

Via the Telegram bot (program for communicating between people and machines), the service technician receives the message from the CPU and responds to it.

The following Figure schematically illustrates the most important components of the solution:

Figure 1-2



### Note

The internet connection can be a wired connection or a wireless connection through public or private communication infrastructure. You can use the appropriate devices from the SCALANCE M-800 series for this purpose.

## 1.2 Principle of operation

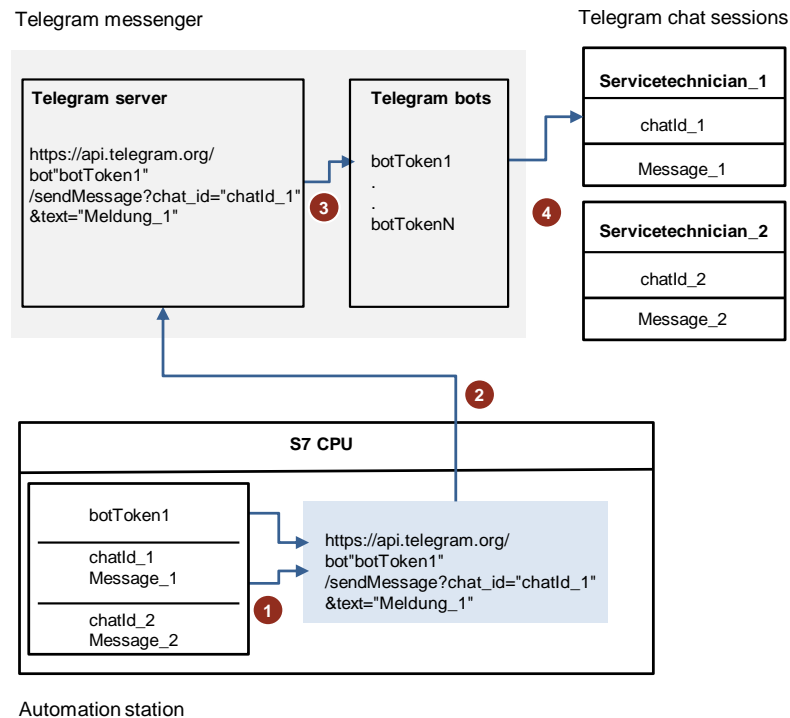
The following functions are implemented in the application example:

- Sending of messages to a service technician via "Telegram Messenger"
- Receipt of a confirmation message from the service technician

### Sending of messages to a service technician via "Telegram Messenger"

The following Figure shows the process for sending a message to the service technician via "Telegram Messenger".

Figure 1-3



1. From the given data, the "LHTTP\_Get" block generates the HTTP request "sendMessage" for sending a message.

The token for sending the message follows the format below:

`https://api.telegram.org/bot"botToken"/sendMessage?chat_id="chatId"&text="Meldung"`

- `api.telegram.org`: URL of the Telegram bot API
- `"botToken"`: String of letters and characters that serves as a "key" for controlling the bot.  
You will receive the token when creating the bot ([chapter 3.1.1](#)).
- `sendMessage`: Methods/command for sending messages to Telegram
- `"chatId"`: Unique identifier of the Telegram chat session with the bot and the service technician ([chapter 3.1.2](#)).
- `"Meldung"` [`"message"`]: message sent to the service technician

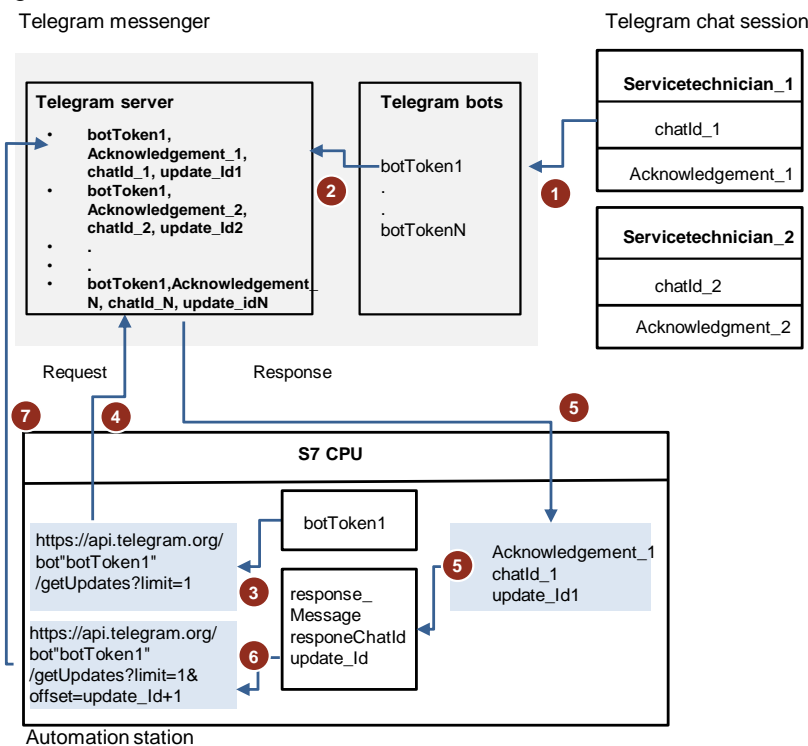
2. The S7 CPU sends the HTTP request to the Telegram server.

3. The Telegram server handles all the encryption and communication with the "Telegram Messenger" via the Telegram bot API.
4. The data are passed to the appropriate "Telegram" recipient (service technician) via the Telegram bot.

### Receive confirmation message from the service technician

The Figure below shows the process for receiving a confirmation message from the service technician.

Figure 1-4



1. The service technician sends a confirmation message to the Telegram bot for the received message.
2. The message is sent to the Telegram server.
3. The "LHTTP\_Get" takes the specified data and creates an HTTP request "getUpdates" for calling the confirmation message.

The token for calling a message from the Telegram server corresponds to the following format:

```
https://api.telegram.org/bot"botToken"/getUpdates?limit=1
```

- api.telegram.org: URL of the Telegram bot API
- "botToken": String of letters and characters that serves as a "key" for controlling the bot.
- getUpdates: Method/command for calling the confirmation messages that were sent to the bot
- limit=1: The first message to be received is always the first to be called from the list on the Telegram server.

You will receive this token when creating the bot ([chapter 3.1.1](#)).

4. The S7 CPU sends the HTTP request to the Telegram server.  
This HTTP request checks whether new messages are available from the Telegram server.
5. The messages are retrieved, analyzed and output to the corresponding parameters of the function block:
  - responseMessage: received confirmation
  - responseChatId: ChatId of the Telegram chat session with the bot and the sender of the confirmation
6. The "LHTTP\_Get" block takes the given data and creates the HTTP request "getUpdates?limit=1&offset=update\_Id+1" for deleting the first message from the list on the Telegram server.

The token for deleting a message from the Telegram server corresponds to the following format:

```
https://api.telegram.org/bot"botToken"/getUpdates?limit=1&offset=update_Id+1
```

- api.telegram.org: URL of the Telegram bot API
- "botToken": String of letters and characters that serves as a "key" for controlling the bot.  
You will receive this token when creating the bot ([chapter 3.1.1](#)).
- getUpdates: Method/command for calling the messages that were sent to the bot
- limit=1: The first message to be received is always the first to be called from the list on the Telegram server.
- update\_Id: The update\_Id is the identifier for each update (e.g. new incoming message, new mentions in a channel, etc.) for the bot.  
The update\_Id is contained in the received message and is evaluated in the block.

7. The S7 CPU sends the HTTP request to the Telegram server.  
The first message is deleted from the list on the Telegram server. Now the message with the update\_id:="update\_Id+1" appears as the first message on the Telegram server.

### Advantages of this solution

- Save time and money when commissioning your own automation solution.
- Reusable STEP 7 code. It can be used as the basis for implementing other messenger services.

Because the structure of the HTTP requests differs by messenger service, a separate block must be created for each messenger service. The code of the FB "TelegramPushNotification" can be used as a basis for this.

#### Note

The library blocks from the "LHTTP" library, part of the Libraries for Communication for SIMATIC Controllers (<https://support.industry.siemens.com/cs/ww/en/view/109780503>), are needed to create the blocks.



## 1.3 Components used

The following hardware and software components were used to create this application example:

Table 1-1

Component	Quantity	Item number	Note
SIMATIC PM1507/1AC/DC24V/3A	1	6EP1332-4BA00	Regulated power supply for SIMATIC S7-1500
SIMATIC S7-1500 CPU 1511-1 PN	1	6ES7511-1AK01-0AB0	A different SIMATIC S7-1500 CPU or SIMATIC S7-1200 CPU of version V4.0 or later can also be used.
SCALANCE MUM856-1	1	6GK5856-2EA00-3DA1	For connecting the controller to the internet. The internet connection can be a wired connection or a wireless connection through public or private communication infrastructure. You can use the appropriate devices from the SCALANCE M-800 series for this purpose.
M12 power cable	1	6XV1801-6D*	For connecting to the power supply
ANT897-4MC	1	6GK5897-4MC00-0AA0	Mobile radio antenna with omnidirectional function for GSM (2G), UMTS (3G) and public 3/4/5G mobile radio networks and private 5G networks worldwide.
Cable - Industrial Ethernet IE TP Cord M12-180/RJ45-180	1	6XV1878-5TH20	For communication between the controller and the SCALANCE MUM856-1
SIM card	1		Micro SIM card
STEP 7 V17 Professional	1	6ES7822-1AA07-0YA5	
Libraries for Communication for SIMATIC Controllers: <a href="https://support.industry.siemens.com/cs/ww/en/view/109780503">https://support.industry.siemens.com/cs/ww/en/view/109780503</a>			For data exchange between a SIMATIC controller via the integrated Ethernet interface with another device in the local network or a Telegram server in the internet over HTTP or HTTPS.
Telegram <a href="https://telegram.org">https://telegram.org</a>	1		Telegram can be downloaded free of charge for Windows, macOS, Linux, iOS and Android.

This application example consists of the following components:

Table 1-2

Component	File name
Project	109763879_Telegram_PushNotification_PROJ_V10.zip <ul style="list-style-type: none"><li>• User name: SiemensUser</li><li>• Password: Siemens123!</li></ul>
Documentation	109763879_Telegram_PushNotification_DOC_V10_en.pdf

## 2 Engineering

### 2.1 Hardware setup

[Chapter 1.3](#) lists the required hardware components.

**CAUTION** The S7-1500 and SCALANCE MUM856-1 installation guidelines must be observed. Please read the corresponding device manuals.

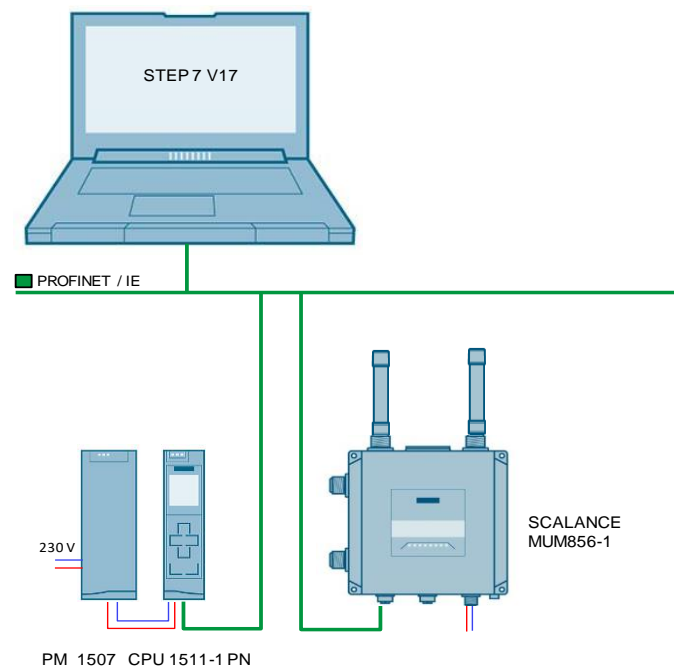
<https://support.industry.siemens.com/cs/ww/en/view/109752841>

<https://support.industry.siemens.com/cs/ww/en/view/109802058>

**CAUTION** Only switch on the power supply after you have completed and checked the assembly!

The following graphic shows the hardware setup of the application.

Figure 2-1



1. Insert the SIM card into the SCALANCE MUM856-1 (<https://support.industry.siemens.com/cs/ww/en/view/109802058>, chapter 5.4).

**CAUTION**

Switch off power supply before changing the SIM card: Before inserting or removing the SIM card, switch off the device's power supply.

Do not open the SIM card tray during operation, otherwise the SIM card and the device may be damaged.

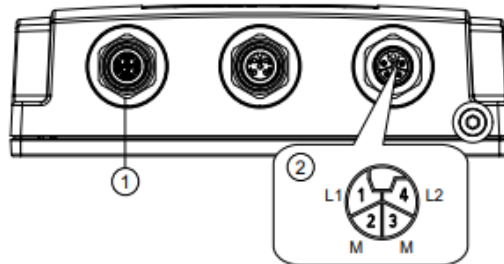
2. Plug the individual hardware components from [Table 1.1](#) into a suitable component rack.
3. Connect the PM 1507 to the power supply. Ensure the polarity is correct.
4. Connect the following devices together:
  - PROFINET interface of the engineering station with the PROFINET interface of the CPU
  - PROFINET interface of the SCALANCE MUM856-1 with the second PROFINET interface of the CPU

### Power supply – position and pinout

There are two options for the power supply:

1. Power over Ethernet via the 8-pin M12 Ethernet interface P1 (position 1). The power supply cannot be connected redundantly.
2. Direct feed via the 4-pin M12 connection socket (position 2). The power supply can be connected redundantly. The L1/L2 inputs are decoupled. There is no distribution of load. The power supply unit with the higher output voltage supplies the device alone.

Figure 2-2



The four-pin M12 socket has the following pinout:

Table 2-1

Pin	Signal	Function
1	L1+	DC 24 V
2	M	GND
3	M	GND
4	L2+	DC 24 V

### Ethernet – position and pinout

The device has an M12 port for connecting to Industrial Ethernet with 10/100/1000 MBit/s: X-coded, 8-pin.

Power over Ethernet can also be supplied through this port.

Figure 2-3

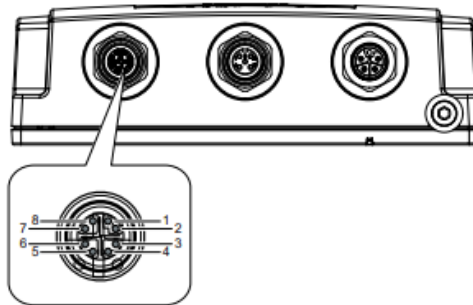


Table 2-2

Pin	Description
1	D0+
2	D0-
3	D1+
4	D1-
5	D3+
6	D3-
7	D2-
8	D2+

The following Table provides an overview of all IP addresses used in this example. Assignment of static IP addresses is assumed.

Table 2-3

Component	IP address	Description
SIMATIC S7 CPU	192.168.0.1	CPU 1511-1 PN
SCALANCE M device	192.168.0.2	SCALANCE MUM856-1
Engineering station	192.168.0.10	STEP 7 V17

The subnet mask in all network components is 255.255.255.0.

#### Note

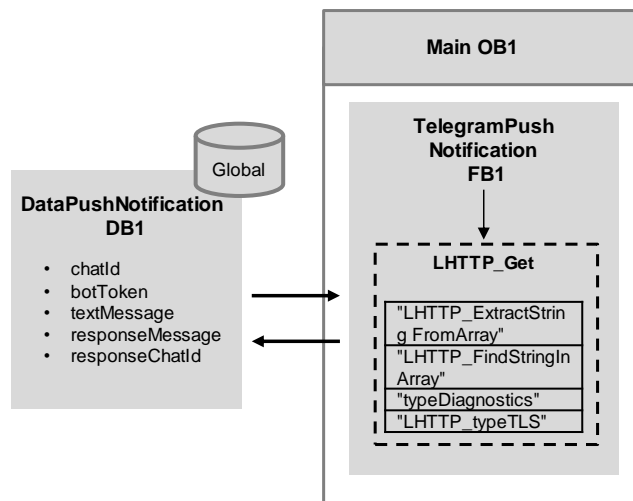
Adjust the IP addresses of the components in your project so that they are in the same subnet.

## 2.2 Engineering of the STEP 7 program

### Diagram

The following graphic shows the program structure of the whole STEP 7 V17 project.

Figure 2-4



### Program blocks

The user program of the SIMATIC S7-1500 CPU consists of the following elements:

Table 2-4

Element	Symbolic name	Description
OB1	Main	In OB1, the "TelegramPushNotification" function block, including the associated instance data block, is called cyclically.
FB1	"TelegramPushNotification"	The "TelegramPushNotification" function block takes the specified data and creates the HTTP request (sendMessage and getUpdates) and sends them to the Telegram server.
DB2	"InstTelegramPushNotification"	Instance data block of the function block "TelegramPushNotification"



Element	Symbolic name	Description
DB1	"DataPushNotification"	Global data block for storing <ul style="list-style-type: none"> <li>the parameters for creating the HTTP requests <ul style="list-style-type: none"> <li>chatId</li> <li>botToken</li> <li>textMessage</li> </ul> </li> <li>the received data <ul style="list-style-type: none"> <li>responseMessage</li> <li>responseChatId</li> </ul> </li> </ul>
Elements from the Libraries for Communication for SIMATIC Controllers	"LHTTP_Get"	This block implements the HTTP method GET to fetch data from the Telegram server.
	"LHTTP_ExtractStringFromArray"	Extracts a string between two specified text parts from an array of chars.
	"LHTTP_FindStringInArray"	Searches an array of chars for a given string.
	"typeDiagnostics"	Data type for storing the error messages that occur in the function block "TelegramPushNotification": <ul style="list-style-type: none"> <li>status: last status code of the interface parameter "status" of the FB</li> <li>subfunctionStatus: Status or returned value from internal instructions/ FBs in which an error occurred. For detailed information, refer to the Online Help on the respective instruction and/or the FB documentation.</li> <li>stateNumber: State of the state machine of the FB in which the error occurred.</li> </ul>
	"LHTTP_typeTLS"	Data type for transferring certificates for secure communication via HTTPS

### 2.2.1 Function block "TelegramPushNotification"

The "TelegramPushNotification" function block takes the specified data and creates the "sendMessage" and "getUpdates" HTTP requests for sending a message, calling a confirmation or deleting unnecessary messages and sends them to the Telegram server.

- With the "sendMessage" HTTP request, messages are sent to the corresponding service technician via the Telegram bot API.
- With the HTTP request "getUpdates?limit=1", the first message that the service technician sent to the bot is the one sent to the controller via the Telegram server. The message is evaluated and the data are output to the parameters "responseMessage" and "responseChatId" of the function block. After evaluating them, the block deletes the message from the list on the Telegram server.

The process repeats until all received messages in the list on the Telegram server have been evaluated and deleted.

- The HTTP request "getUpdates?limit=1&offset=update\_Id+1" causes the first message in the list of messages that were sent to the bot in the last 24 hours to be deleted without being evaluated.

The process repeats until all received messages in the list on the Telegram server have been deleted.

#### Send messages to a service technician with "Telegram Messenger"

This functionality is implemented as a sequencer with the following states:

Figure 2-5

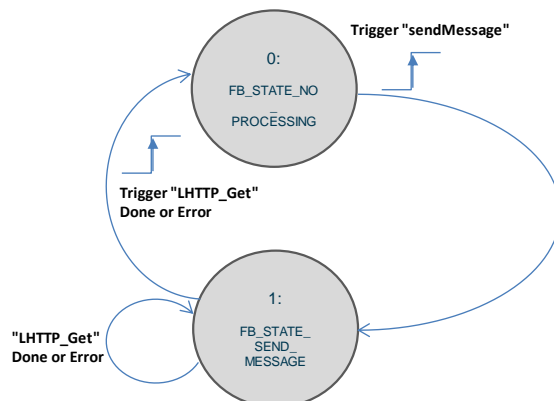


Table 2-5

State		Description
0	FB_STATE_NO_PROCESSING	<ul style="list-style-type: none"> <li>• Idle process: Process is doing nothing.</li> <li>• Waiting for rising edge at the "sendMessage" input</li> </ul>
1	FB_STATE_SEND_MESSAGE	<ul style="list-style-type: none"> <li>• Calls the library block "LHTTP_GET" for sending the HTTP request "sendMessage" to the Telegram server.</li> <li>• After successfully sending the request or in case of error, back to state "FB_STATE_NO_PROCESSING"</li> </ul>

### Receive confirmation from service technician

This functionality is implemented as a sequencer with the following states:

Figure 2-6

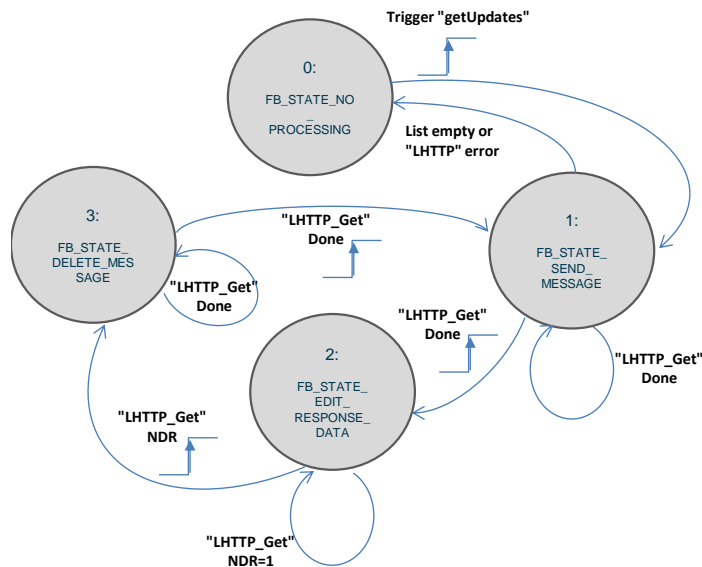


Table 2-6

State		Description
0	FB_STATE_NO_PROCESSING	<ul style="list-style-type: none"> <li>Idle process: Process is doing nothing.</li> <li>Waiting for rising edge at the "getUpdates" input</li> </ul>
1	FB_STATE_SEND_MESSAGE	<ul style="list-style-type: none"> <li>Calls the library block "LHTTP_GET" for sending the HTTP request "getUpdates?limit=1" to the Telegram server.</li> <li>If no confirmations are present, or in case of an error, return to state "FB_STATE_NO_PROCESSING"</li> </ul>
2	FB_STATE_EDIT_RESPONSE_DATA	After sending the HTTP request "getUpdates?limit=1", the first message from the list on the Telegram server is evaluated and output to the corresponding parameters of the FB.
3	FB_STATE_DELETE_MESSAGE	After evaluating the received message, the block deletes this message from the list on the Telegram server. <ul style="list-style-type: none"> <li>Calls the library block "LHTTP_GET" for deleting the message.</li> <li>After successfully sending the request, back to state "FB_STATE_SEND_MESSAGE"</li> </ul>

## Delete list of received messages without evaluation

**Note**

This step is necessary in order to remove unnecessary messages sent to the bot within the last 24 hours.

This functionality is implemented as a sequencer with the following states:

Table 2-7

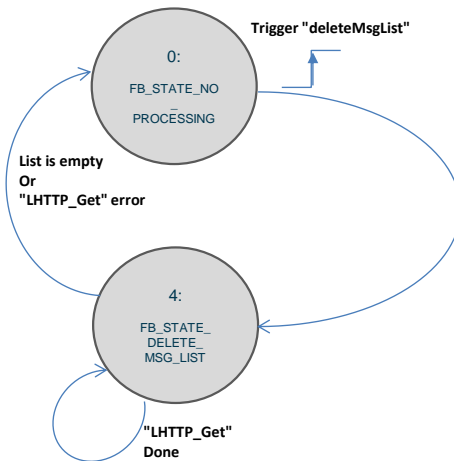


Table 2-8

State		Description
0	FB_STATE_NO_PROCESSING	<ul style="list-style-type: none"> <li>Idle process: Process is doing nothing.</li> <li>Waiting for rising edge at input "deleteMsgList"</li> </ul>
4	FB_STATE_DELETE_MSG_LIST	<ul style="list-style-type: none"> <li>An HTTP request "getUpdates?limit=1&amp;offset=update_Id+1" is sent to the Telegram server.</li> <li>The first message is deleted from the list on the Telegram server.</li> <li>The process repeats until the list on the Telegram server is empty.</li> <li>If no messages are present, or in case of an error, return to state "FB_STATE_NO_PROCESSING"</li> </ul>

## Parameters

The Figure and Table below show the call interface of the function block "TelegramPushNotification".

Figure 2-7

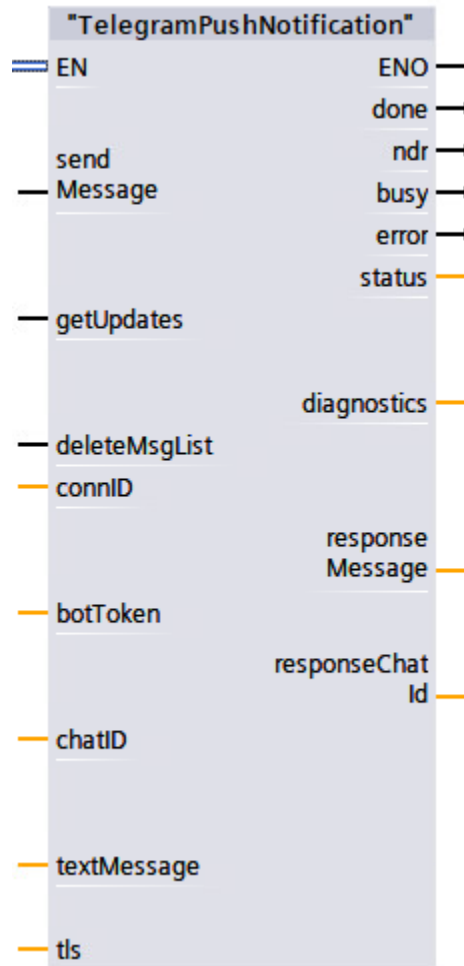


Table 2-9

Symbol	Data type	Description
sendMessage	BOOL	<ul style="list-style-type: none"> <li>A rising edge initiates the connection to the Telegram server and the HTTP request "sendMessage" is sent.</li> <li>The messages are passed to the corresponding "Telegram" recipient ("chatID") via the Telegram bot API.</li> <li>The process is only activated if the inputs "deleteMsgList" and "getUpdates" are FALSE.</li> </ul>

Symbol	Data type	Description
getUpdates	BOOL	<ul style="list-style-type: none"> <li>A rising edge initiates the connection to the Telegram server and the HTTP request "getUpdates" is sent.</li> <li>The messages sent to the bot in the last 24 hours are forwarded to the controller via the Telegram bot API.</li> <li>The messages are evaluated and output to the parameters "responseMessage" and "responseChatId" of the FB.</li> <li>After evaluating them, the block deletes the messages from the list on the Telegram server.</li> <li>The process is only activated if the inputs "deleteMsgList" and "sendMessage" are FALSE.</li> </ul>
deleteMsgList	BOOL	<ul style="list-style-type: none"> <li>A rising edge causes the list of messages sent to the bot in the last 24 hours to be deleted without evaluation.</li> <li>The process is only activated if the inputs "sendMessage" and "getUpdates" are FALSE.</li> </ul>
connID	CONN_OUC	Unique connection ID
botToken	STRING	<ul style="list-style-type: none"> <li>API token: String of letters and characters that serves as a "key" for controlling the bot.</li> <li>This field may not be left blank.</li> </ul> <p><b>Note:</b> You will receive this API token when creating the bot (<a href="#">chapter 3.1.1</a>). This API token allows anybody to access and modify the bot. You should make a note of it and not reveal it to others.</p>
chatID	STRING	<ul style="list-style-type: none"> <li>Unique identifier of the Telegram chat session with the bot and the recipient of the message (<a href="#">chapter 3.1.2</a>)</li> <li>This field in the HTTP request "sendMessage" must not be left blank.</li> </ul>
textMessage	STRING [247]	<ul style="list-style-type: none"> <li>Sent message (max. length: 247 characters)</li> <li>This field in the HTTP request "sendMessage" must not be left blank.</li> </ul>



Symbol	Data type	Description
tls	"LHTTP_typeTLS"	<p>TLS certificates for secure data transmission (HTTPS)</p> <p>For unsafe data transmission (HTTP) leave unconnected.</p> <p><b>Note:</b></p> <p>Information about the PLC data type "LHTTP_typeTLS" can be found in chapter 3.7 of the library description "<a href="#">Library Comm Controller</a>".</p>
done	BOOL	<ul style="list-style-type: none"> <li>TRUE: Job complete.</li> <li>If "deleteMsgList", "sendMessage" or "getUpdates" are reset, "done" is also reset.</li> </ul>
ndr	BOOL	<ul style="list-style-type: none"> <li>New message/confirmation was received and stored in the memory range.</li> <li>Only TRUE for one cycle.</li> </ul>
busy	BOOL	<ul style="list-style-type: none"> <li>TRUE: Job is being executed.</li> </ul>
error	BOOL	<ul style="list-style-type: none"> <li>TRUE: An error has occurred.</li> <li>The error must be remedied by the user.</li> <li>If "deleteMsgList", "sendMessage" or "getUpdates" are reset, "error" is also reset.</li> </ul>
status	WORD	<ul style="list-style-type: none"> <li>Status of the instruction</li> <li>Error information ("error"=true) (<a href="#">chapter 2.7</a>)</li> </ul>
diagnostics	"typeDiagnostics"	<p>Advanced diagnostic information:</p> <ul style="list-style-type: none"> <li>status: Last status code of the interface parameter "status" of the FB</li> <li>subfunctionStatus: Status or returned value from internal instructions or FBs where an error occurred. For detailed information, refer to the Online Help for the instruction in question, or the documentation of the FB.</li> <li>stateNumber: State of the FB's state machine in which the error occurred.</li> </ul>
responseMessage	STRING	The last received message
responseChatId	STRING	Unique identifier of the Telegram chat session with the bot and the sender of the last message/confirmation message

**Note**

Only the data of the last received message are output at the parameters "responseMessage" and "responseChatId".

To avoid being overwritten by new data, make sure that the received data are saved immediately after the "ndr" parameter is set.

## 2.2.2 Global data block "DataPushNotification"

The global data block "DataPushNotification" contains the following tags:

Figure 2-8

TelegramPushNotification ▶ PLC_1 [CPU 1511-1 PN] ▶ Program blocks ▶ DataPushNotification [DB1]		
DataPushNotification		
	Name	Data type
1	Static	
2	deleteMsgList	Bool
3	sendMessage	Bool
4	getUpdates	Bool
5	chatId	String
6	botToken	String
7	textMessage	String[247]
8	tls	*LHTTP_typeTLS*
9	validateServerIdentity	Bool
10	serverCert	UDInt
11	clientCert	UDInt
12	responseMessage	String
13	responseChatId	String
14	diagnostics	*typeDiagnostics*
15	status	Word
16	subfunctionStatus	DWord
17	stateNumber	DInt

1. The PLC tags for calling the function block "TelegramPushNotification" in OB1:

- deleteMsgList  
Activates deletion of the list of messages on the Telegram server that were sent to the bot.
- sendMessage  
Activates sending of the HTTP request "sendMessage".
- getUpdates  
Activates sending of the HTTP request "getUpdates".

2. The PLC tags for creating the HTTP requests:

- chatId  
Unique identifier of the Telegram chat session with the bot and the recipient of the message.
- botToken  
String of letters and characters that serves as a "key" for controlling the bot.
- textMessage  
Sent message/notification
- tls  
TLS certificates ([chapter 2.3.2](#)) for secure data transmission (HTTPS).

3. The PLC tags for saving the received data:

- responseMessage  
Received message/confirmation
- responseChatId  
Unique identifier of the Telegram chat session with the bot and the sender of the confirmation.

### 4. The PLC data type "typeDiagnostics"

The "diagnostics" output parameter of the "TelegramPushNotification" function block helps store error information and requires the "typeDiagnostics" data type as a PLC data type:

- status:  
Last status code of the interface parameter "status" of the FB.
- subfunctionStatus:  
Status or returned value from internal instructions or FBs where an error occurred. For detailed information, refer to the Online Help for the instruction in question, or the documentation of the FB.
- stateNumber:  
State of the FB's state machine in which the error occurred.

## 2.3 Configuring the SIMATIC S7 CPU

This chapter describes the most important project engineering steps.

### Note

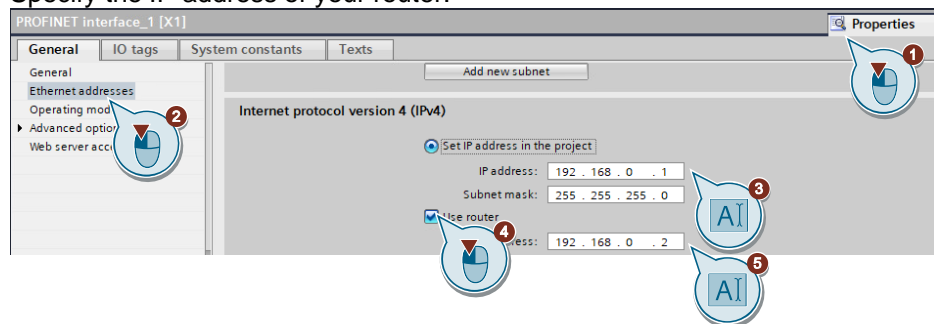
The configuration is fully implemented in the example project. This chapter is for information only.

### 2.3.1 Configure the Ethernet interface

#### Configure router

To run HTTP communication across subnet boundaries, e.g. via the internet, you must specify the IP address of your router in the device properties of your PLC. Proceed as follows:

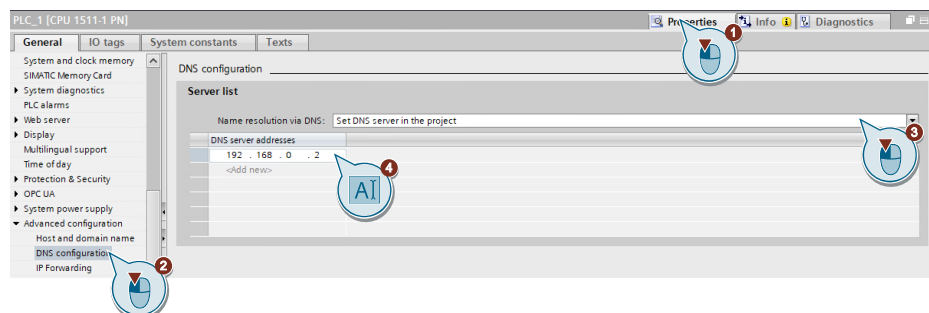
1. Open the device properties of your PLC.
2. Open the "PROFINET interface [Xx] > Ethernet addresses" area of the PROFINET interface that you want to use for HTTP communication.
3. Enter the IP address of the S7 CPU according to [Table 2-3](#).
4. Tick the checkbox for "Use router".
5. Specify the IP address of your router.



#### Configure the DNS server

To be able to use and resolve domain names in the URL, you must specify at least one DNS server in the device properties of your PLC. Often the router is also a DNS server. Proceed as follows:

1. If necessary, open the device properties of your PLC.
2. Open the "DNS configuration" area.
3. Set the DNS server in the project.
4. Save the IP address of at least one DNS server.



### 2.3.2 Certificate management

With HTTPS, the user data is transmitted in encrypted form. Web server and client (in this case the PLC) exchange certificates with each other before data transfer. In order for the PLC to confirm the authenticity of the web server, it must know the root certificate from which the web server certificate is derived.

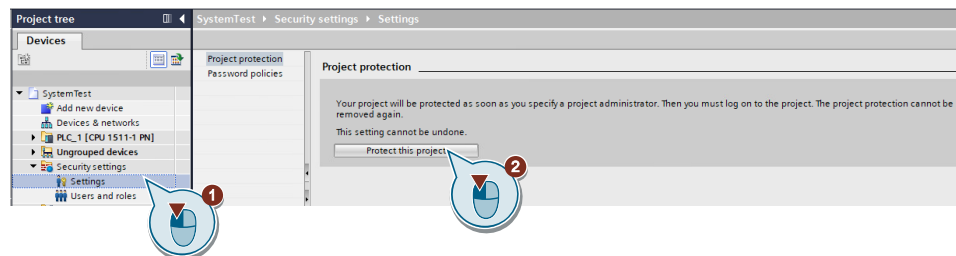
In TIA Portal, the certificates are managed in a global certificate manager. The certificate manager contains an overview of all certificates used in the project. In the certificate manager, for example, you can import new certificates and export, renew, or replace existing certificates. Each certificate is assigned an ID that can be used to reference the certificate in the program blocks.

Perform the following steps to transmit data securely with HTTPS:

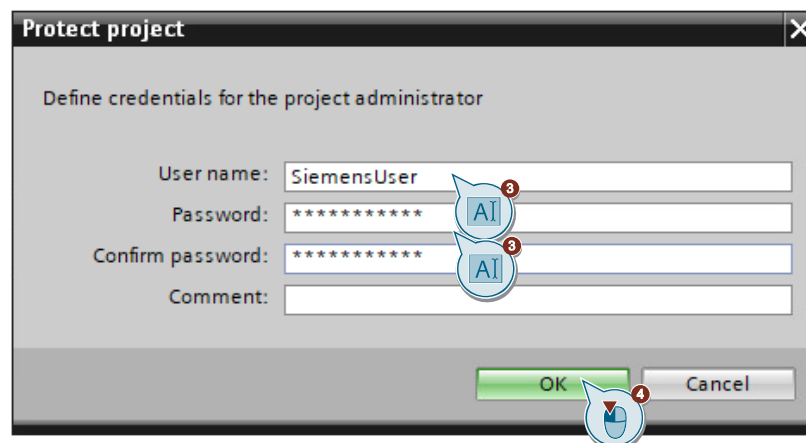
#### Protect TIA Portal project with a password

In order to manage certificates in TIA Portal, you must protect your project with a password. Proceed as follows:

1. Open the section "Security settings > Settings" from the project tree.
2. Click "Protect this project".



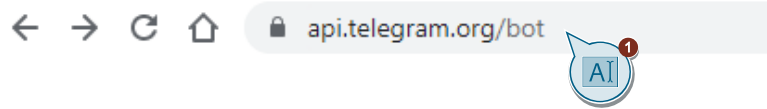
3. Assign a user name and password.
4. Click "OK".



### Download certificate from Telegram server and import in TIA Portal

Proceed as follows to download the certificate from a webpage. In the following the procedure with Google Chrome is described.

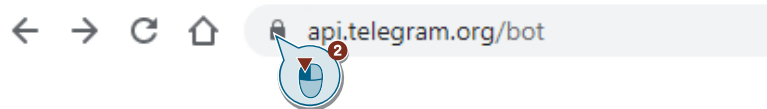
1. Open the desired web page in a web browser (<https://api.telegram.org/bot>).



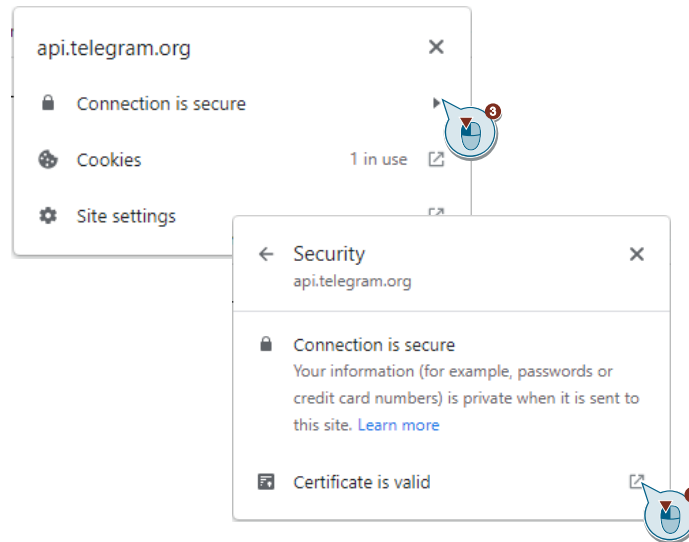
#### Note:

Ignore the notifications shown on this webpage.

2. Click the padlock icon in the address bar.

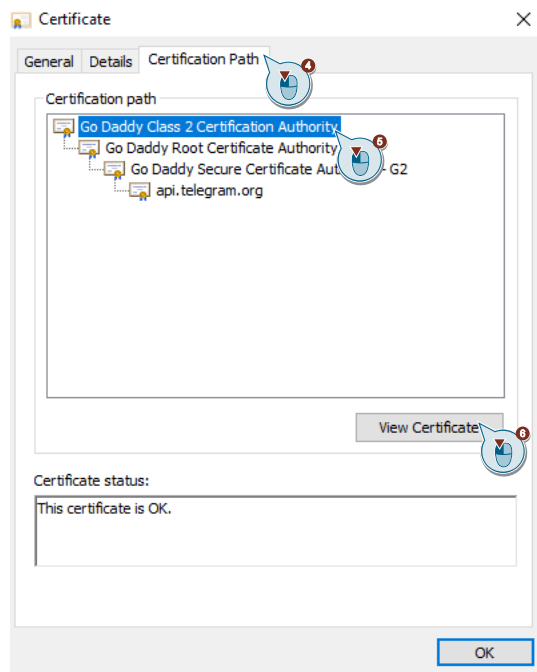


3. Click on "Certificate (Valid)".

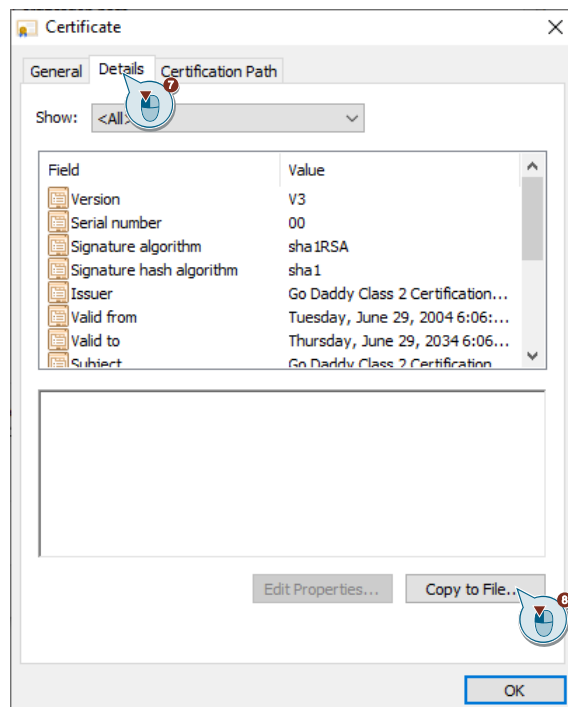




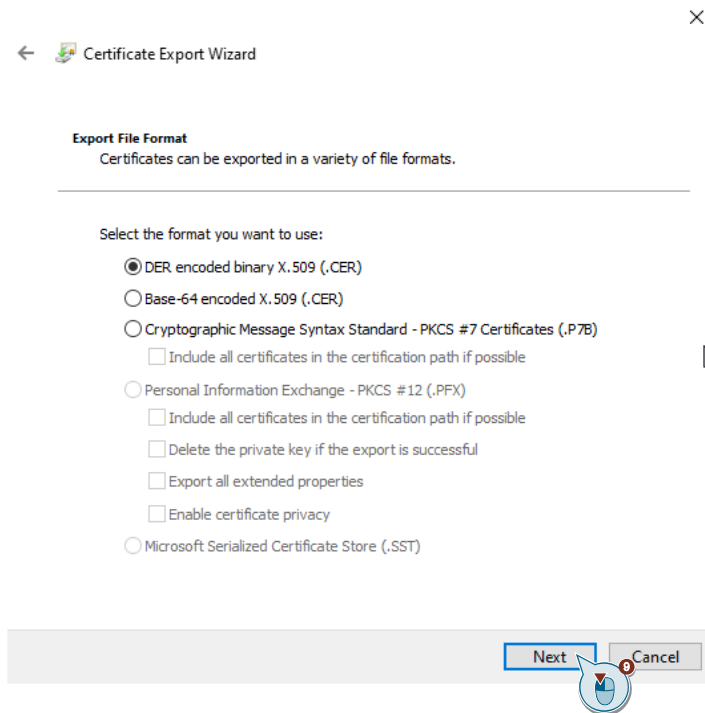
4. Open the "Certification Path" tab.
5. Select the topmost certificate (root certificate from which the others are derived).
6. Click "View Certificate".



7. Open the "Details" tab.
8. Click on "Copy to File..."

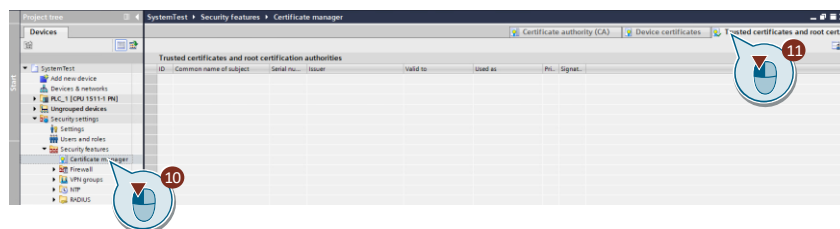


## 9. Save the DER-encoded certificate.



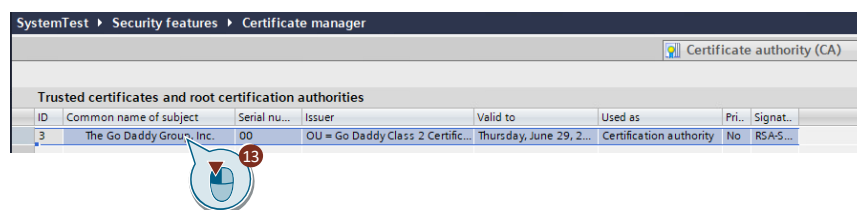
## 10. From the TIA Portal project tree, open the "Security settings &gt; Security features &gt; Certificate manager" section.

## 11. Open the "Trusted certificates and root certification authorities" tab.



## 12. Right-click in the workspace and select "Import".

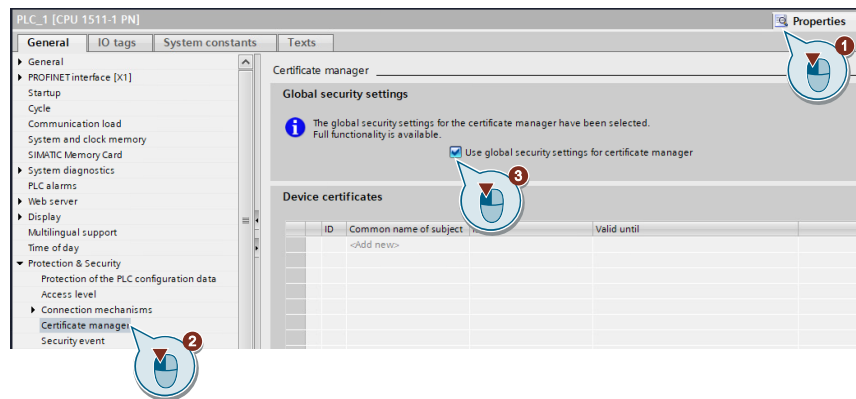
## 13. Import the certificate you saved from the Telegram server.



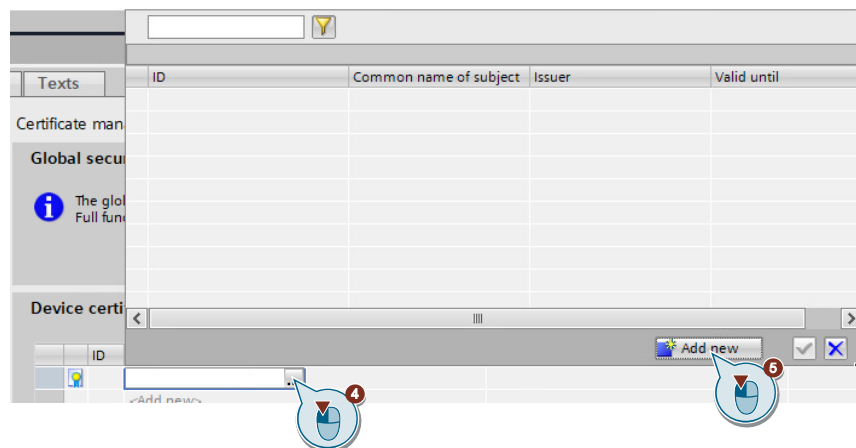
### Create device certificate

For the encrypted HTTPS connection, the client (in this case, the PLC) also needs a certificate. Proceed as follows to create a device certificate:

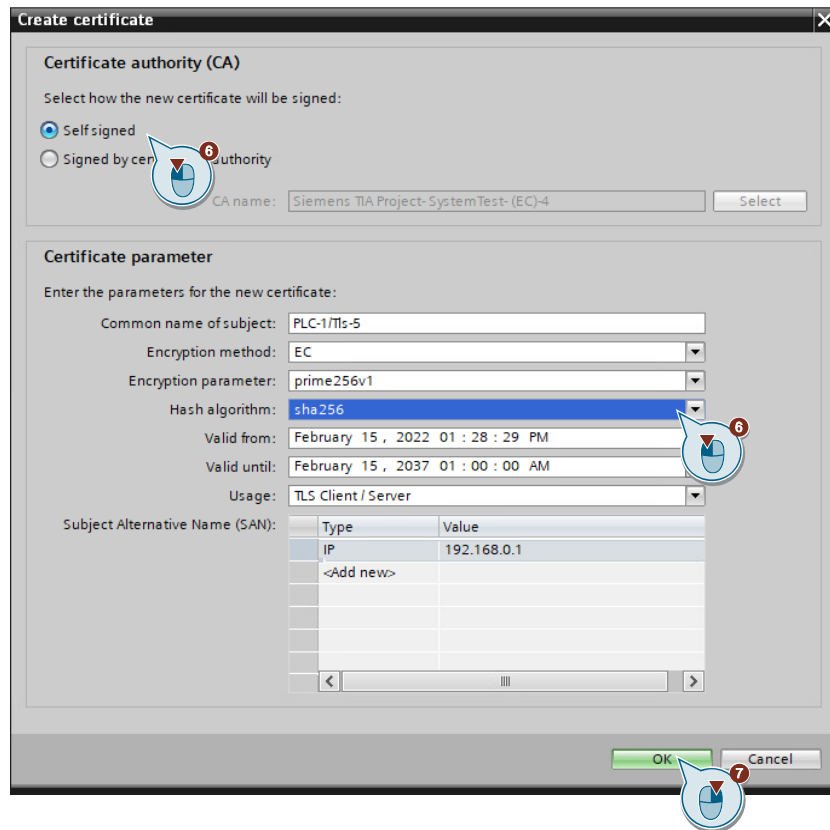
1. Open the device properties of your PLC.
2. Open the "Protection & Security > Certificate manager" section.
3. To use the imported certificate from the global certificate manager and derive a device certificate from the root certificate of the TIA Portal project, tick the checkbox labeled "Use global security settings for certificate manager".



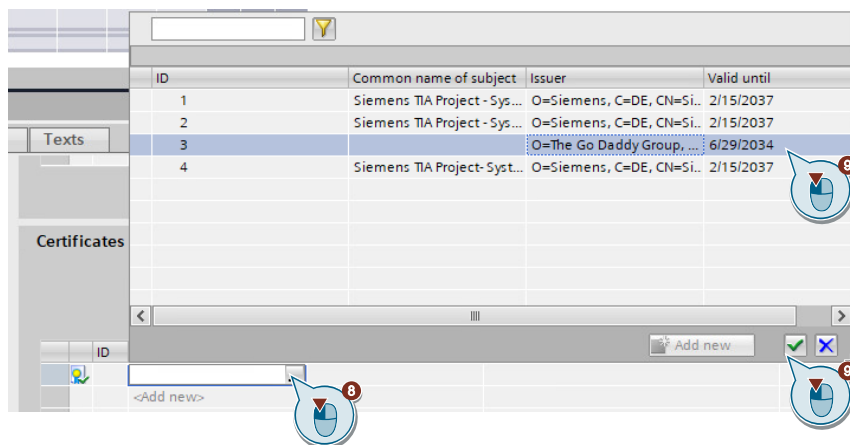
4. Under "Device certificates", double-click the "<Add new>" cell.
5. Click on the same cell again and then on "Add new".



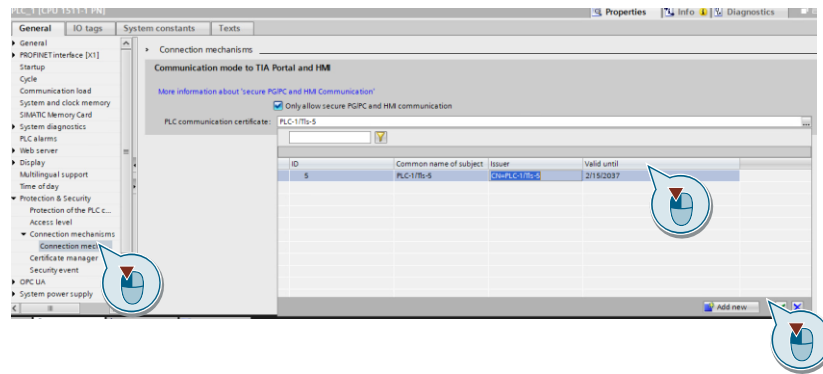
6. Select "Self signed" and the "sha256" encryption algorithm for maximum compatibility.
7. Confirm with "OK".



8. Under "Certificates of the partner devices", double-click the cell "<Add new>".
9. Click the same cell again and select the Telegram server's imported root certificate.



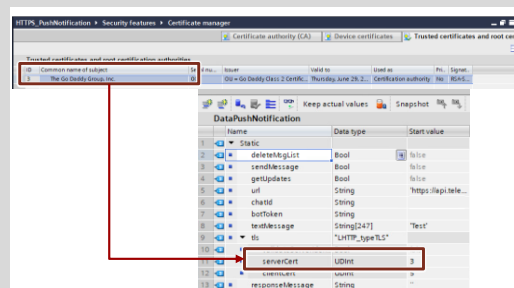
10. Select the imported root certificate of the Telegram server under "Connection mechanisms > Communication mode to TIA Portal and HMI".



The ID of the server certificate can be found in the global certificate manager under "Security settings > Security features > Certificate manager > Trusted certificates and root certification authorities".

You will use the ID of the server certificate at the "serverCer" parameter of the PLC data type "LHTTPS\_typeTLS" in the DB "DataPushNotification".

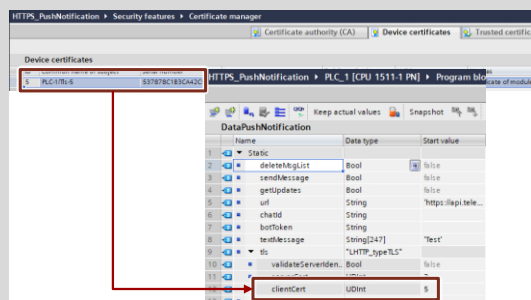
**Note**



The ID of the device certificate is located in the global certificate manager under "Security settings > Security features > Certificate manager > Device certificates".

You will use the ID of the device certificate at the "clientCer" parameter of the PLC data type "LHTTPS\_typeTLS" in the DB "DataPushNotification".

**Note**



## 2.4 Configuring the SCALANCE MUM856-1

To connect the S7 CPU to the internet via the cellular radio network, you must make the following settings in the Web Based Management page of the SCALANCE MUM856-1:

- Enable mobile radio interface
- Enter valid PIN for the inserted SIM card
- Enable firewall / define IP rules
- Modify IP address

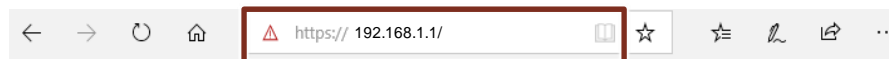
### Note

General information on configuring the SCALANCE MUM856-1 can be found in the configuration manual for the SCALANCE M-800 devices.

<https://support.industry.siemens.com/cs/ww/en/view/109751635>

The SCALANCE devices are configured in Web Based Management:

1. Open an internet browser and enter the IP address of the SCALANCE MUM856-1 in the address bar.

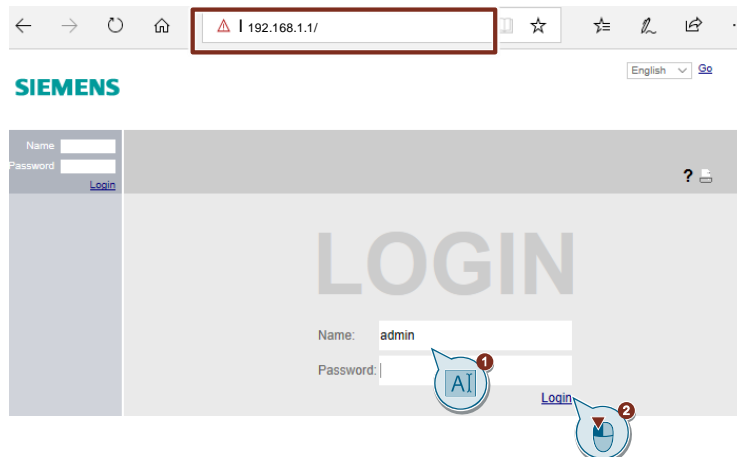


### Note

When you log in for the first time, enter the IP address 192.168.1.1.

The engineering station must be in the same subnet as the SCALANCE MUM856-1.

2. Log in as an administrator.



### Note

When you log in for the first time, you will be prompted to change the password for the "admin" user.



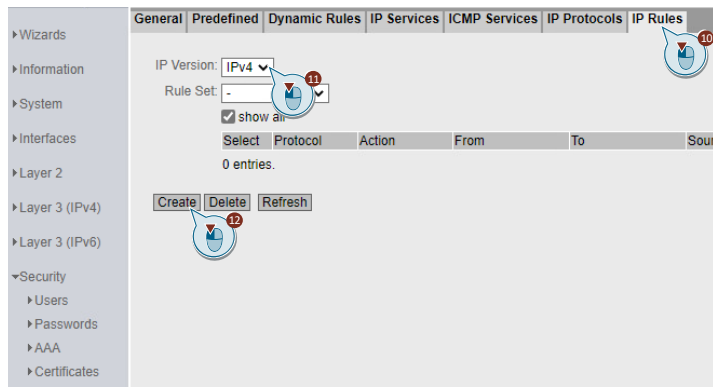
3. Switch to the "Interfaces > Mobile" menu.
4. Enable the mobile radio interface ("Enable Mobile Networks Interface").
5. Enter the PIN of the inserted SIM card.
6. Apply the settings ("Set Values").

### Note

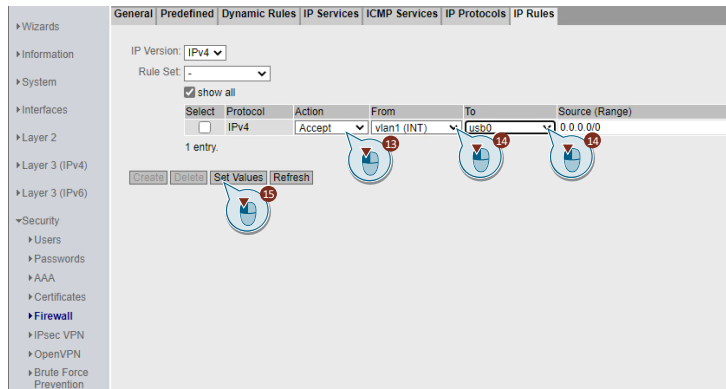
You can obtain the PIN from your mobile network operator.  
The device also works with PIN-less SIM cards. In this case, leave the field blank.

7. Switch to the menu "Security > Firewall".
8. Enable the firewall.
9. Apply the settings ("Set Values").

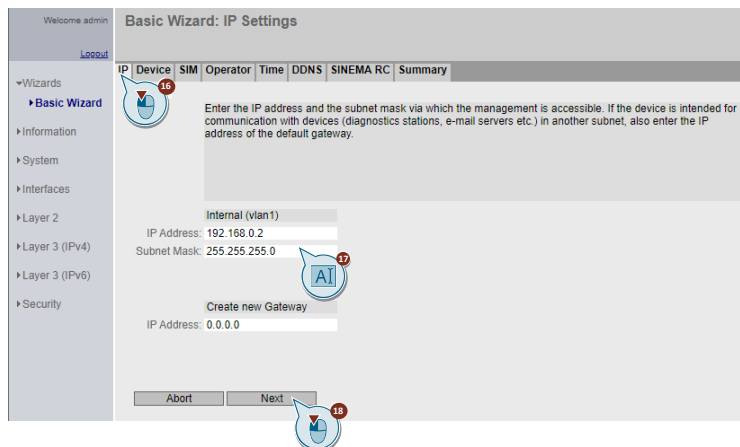
10. Switch to the menu "Security > Firewall > IP Rules".
11. Define the IP version that the firewall will apply for.
12. Click "Create".



13. As the action, select "Accept". The data packets will be allowed through.
14. Define the communication direction for the IP rule (from VLAN1... to WAN).
15. Apply the settings ("Set Values").



16. Switch to the "Wizards > Basic Wizards > IP" menu.
17. Modify the IP address of the device ([Table 2-3](#)).
18. Apply the settings ("Next > Next>...> Set Values").



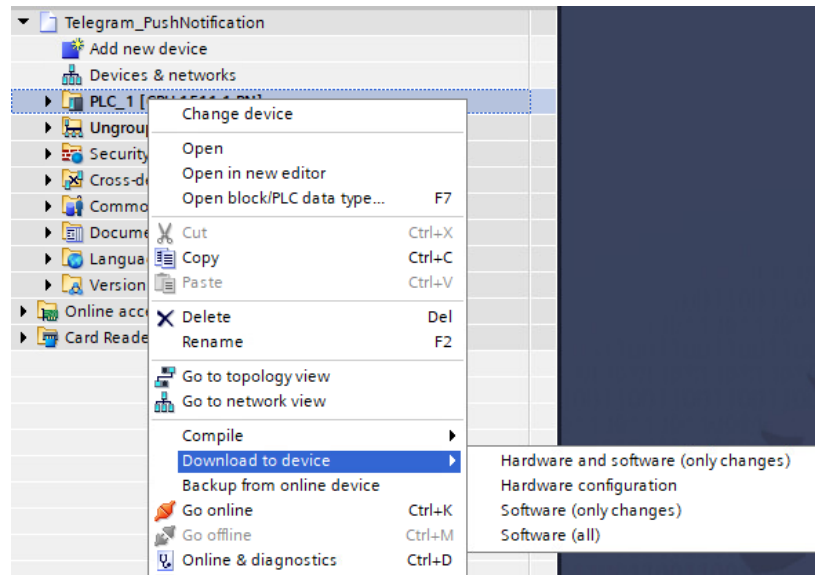
19. Modify the IP address of the engineering again so that all components are located in the same subnet ([Table 2-3](#)).

## 2.5 Downloading the STEP 7 V17 project

Proceed as follows to download the STEP 7 program:

1. Extract the STEP 7 V17 project  
"109753879\_Telegram\_PushNotification\_PROJ\_V10.zip".
2. Start TIA Portal V17.
3. Open the project "Telegram\_PushNotification.ap17".
4. Log in with the user credentials:
  - User name: SiemensUser
  - Password: Siemens123!
5. Connect the Ethernet port of the programming computer with the Ethernet port of the S7-1500 CPU.
6. Load the configuration "PLC\_1".

To do this, right-click the device in the project tree and select the menu "Download to device > Hardware and software (only changes)".



## 2.6 Operation

### Introduction

In this chapter, you will learn how to operate the following functions, which are part of the application example

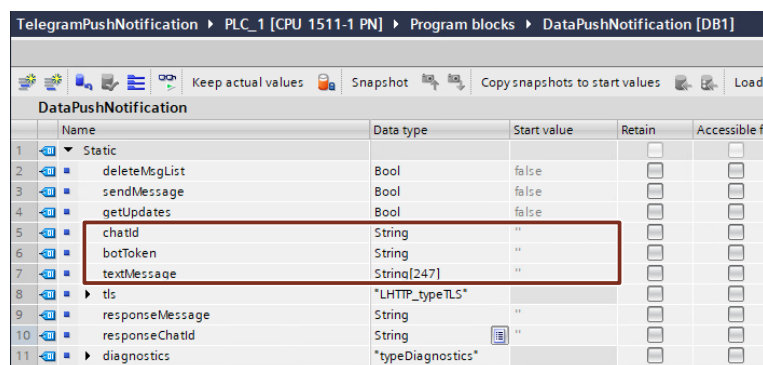
- Send messages to a service technician with "Telegram Messenger"
- Receive and evaluate confirmations from a service technician.

### Requirements

- The Telegram bot has been created (see [chapter 3.1.1](#))
- The Telegram chat session with the bot has been started (see [chapter 3.1.2](#))
- The chat ID of the Telegram chat session is known (see [chapter 3.1.2](#))

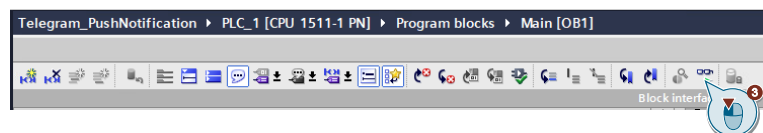
### Procedure

1. Open the DB "DataPushNotification" and populate the parameters with values:
  - chatId:  
Unique identifier of the Telegram chat session with the bot and the service technician
  - botToken:  
API token of the bot you created
  - textMessage:  
Desired message. This field must not remain blank.

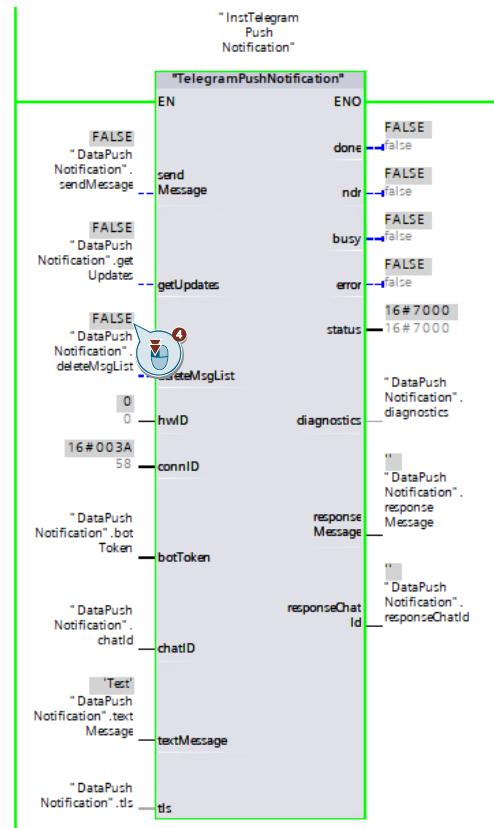


TelegramPushNotification ▶ PLC_1 [CPU 1511-1 PN] ▶ Program blocks ▶ DataPushNotification [DB1]					
	Name	Data type	Start value	Retain	Accessible f..
1	Static				
2	deleteMsgList	Bool	false	<input type="checkbox"/>	<input type="checkbox"/>
3	sendMessage	Bool	false	<input type="checkbox"/>	<input type="checkbox"/>
4	getUpdates	Bool	false	<input type="checkbox"/>	<input type="checkbox"/>
5	chatId	String	"	<input type="checkbox"/>	<input type="checkbox"/>
6	botToken	String	"	<input type="checkbox"/>	<input type="checkbox"/>
7	textMessage	String[247]	"	<input type="checkbox"/>	<input type="checkbox"/>
8	tls	*LHTTP_typeTLS*		<input type="checkbox"/>	<input type="checkbox"/>
9	responseMessage	String	"	<input type="checkbox"/>	<input type="checkbox"/>
10	responseChatId	String	"	<input type="checkbox"/>	<input type="checkbox"/>
11	diagnostics	*typeDiagnostics*		<input type="checkbox"/>	<input type="checkbox"/>

2. Save and load the changes.
3. Open and activate OB1.



4. In Network 1, select the "deleteMsgList" input of "TelegramPushNotification" to delete the entire list of messages on the Telegram server that were sent to the bot in the last 24 hours.



### Result:

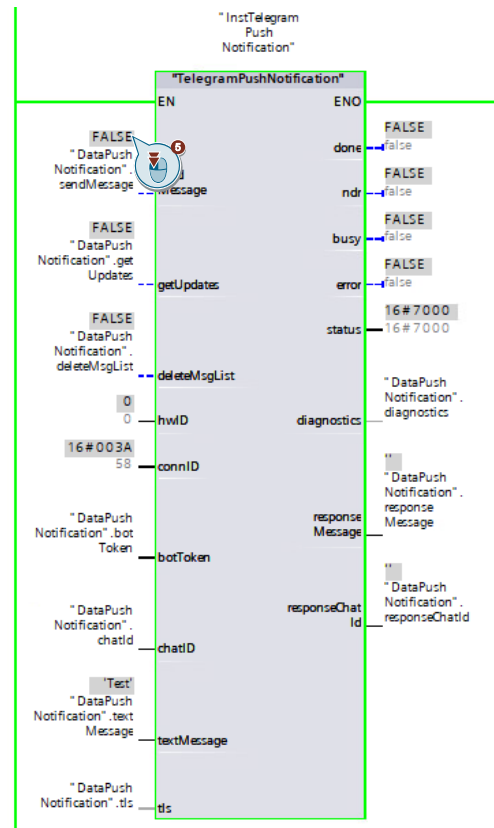
The list with the messages has been fully deleted.

Output "done": TRUE

### Note

This step is necessary in order to remove unnecessary messages sent to the bot within the last 24 hours that are on the Telegram server.

5. In Network 1, deactivate the "deleteMsgList" input and activate the "sendMessage" input of "TelegramPushNotification" in order to send a message to the Telegram recipient.



### Result:

The message has been sent to the desired recipient.

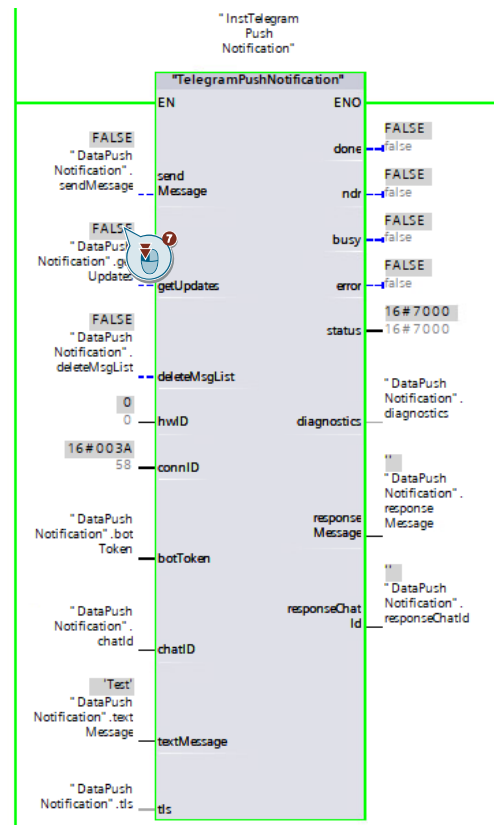
Output "done": TRUE

6. Send any confirmation messages to the bot via "Telegram Messenger".

**Note**

Chapters [3.1.1](#) and [3.1.2](#) will demonstrate how to create a bot and send messages to it.

7. In Network 1, deactivate the "sendMessage" input and activate the "getUpdates" input of "TelegramPushNotification" in order to retrieve the received messages from the Telegram server.

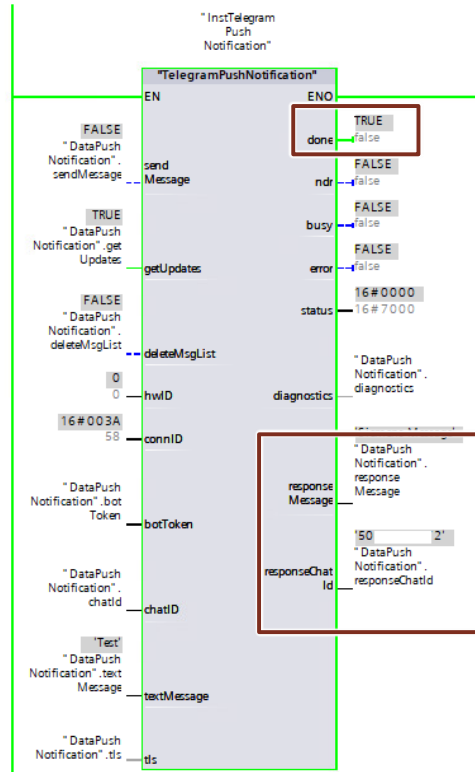


**Result:**

The messages have been received and evaluated.

Output "done": TRUE

Only the data of the last received message are output at the parameters "responseMessage" and "responseChatId". Make sure that the old data are backed up before they are overwritten by new data.





## 2.7 Troubleshooting

The "TelegramPushNotification" function block provides the status of the instruction via the "status" output parameter. In the event of an error, it sends various error messages via the output parameters "status" and "diagnostics".

### Status of the instruction

The following Table provides an overview of the instruction status.

Table 2-10

"status"	Meaning
16#0000	The job finished with no errors.
16#7000	No job processing active.
16#7001	Initial call when job is started.
16#7002	Second call when job is started.

### Error messages

The following Table provides an overview of the error messages from the "TelegramPushNotification" function block.

Table 2-11

"status"	Meaning
16#8401	The input parameters "sendMessage" and/or "getUpdates" and/or "deleteMsgList" of the FB "TelegramPushNotification" were triggered at the same time.
16#8402	The parameter "botToken" has no values.
16#8403	The parameters "chatID" and "textMessage" have no values.
16#8600	Error message due to an undefined state in the state machine.
16#8701	"LHTTP_GET" error The status or returned value of the internal instruction can be found in the tag <diagnostics.subfunktionStatus>.

Information on the status outputs and for diagnosing the block "LHTTP\_Get" can be found in chapter 3.8 of the description of the library "Library\_Comm\_Controller".

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

## 3 Useful information

### 3.1 Telegram Messenger

#### Introduction

"Telegram Messenger" stands out from competing services mainly thanks to its unique features for bots and channels. A Telegram bot is a good option for delivering information from practically any place imaginable whenever that information is needed.

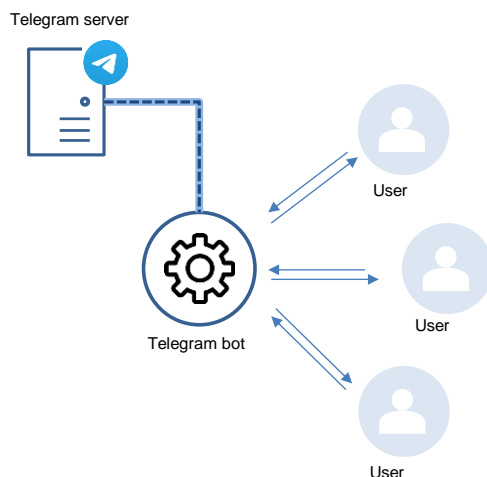
Telegram does not just facilitate communication between people, its bots also allow for communication between man and programmable machines. Accordingly, you can send messages to multiple devices or receive messages from them with one click.

#### Functions

Messages, commands and requests that users send are forwarded to the software running on your servers. The intervening Telegram server handles all the encryption and communication with the Telegram API.

You communicate with the Telegram server via a simple HTTPS interface, the Telegram bot API. The Telegram bot API was developed for the creation of bots.

Figure 3-1



A Telegram bot (short for "robot") is a program that behaves like an ordinary chat partner with some additional functions. It can accomplish specified tasks independently and without user intervention.

A Telegram bot can do virtually anything a human chat partner can do. It can send you the following information either automatically or upon request:

- Text messages
- Images
- Videos
- Files of any other kind

An important feature of a Telegram bot is to run commands in a Telegram chat which can then directly trigger actions or query information.

### 3 Useful information

---

You converse with a Telegram bot with text commands, defined when the bot was created/programmed, that always begin with "/".

In principle, it is possible to use any programming language that runs on a server and that can respond to queries via HTTPS.

All queries (requests) must be presented in the following form:

`https://api.telegram.org/bot<botToken>/METHOD_NAME`

You can obtain more information about the methods from the following link:

<https://core.telegram.org/bots/api>

### 3.1.1 Creating a Telegram bot

The "Telegram" messenger service provides an app for Android/iOS smartphones that you can use to create your own bots.

You can utilize various commands to control bots in your chat or in group chats. It is possible to program whether bots in group chats have read access to all messages or only the ones that contain "@Botname".

Proceed as follows to create a Telegram bot:

1. Download the Telegram application ([Telegram application](#)).
2. Sign in to Telegram.

#### Note

You sign up with telegram using a phone number, but the number does not have to be the number of the device itself. The telephone number is then verified by SMS or call.

Only mobile numbers are allowed (no landlines).

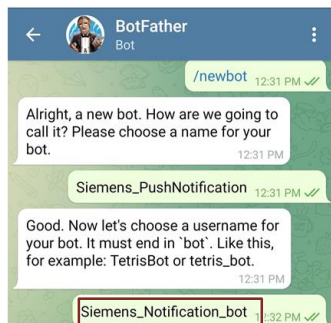
3. Enter the BotFather in the search field of the Telegram client and start a chat with him.
4. Enter the command \newbot to create a new bot.



5. Enter a name for the newly created bot.



6. Enter a user name for the newly created bot.



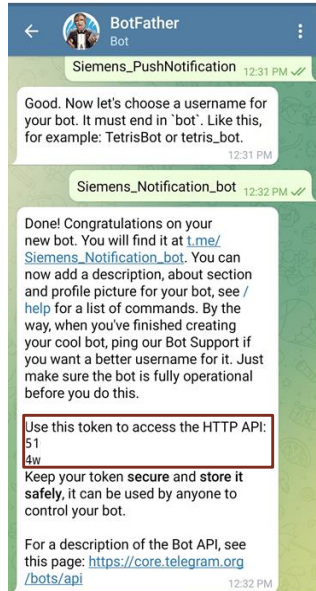
#### Note

The user name will help you find the bot later. The name must end in "\_bot".

#### Result:

The BotFather responds with a long string of letters and characters that serve as a "key" for controlling the bot and which allow access to the HTTP API – the API token.

This API token allows anybody to access and modify the bot. You should make a note of it and not reveal it to others.



7. Edit the privacy settings of the newly created bot by entering the command \mybots.

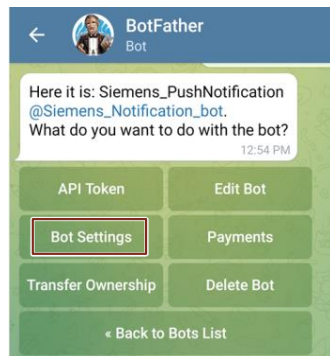


8. Select the bot you wish to edit.



9. Click on "Bot Settings" to edit the bot's privacy settings.

- Allow Groups? –  
If the bot will be added to a group, groups must be allowed.
- Group Privacy  
Must be activated if the bot also needs to receive messages from group chats.

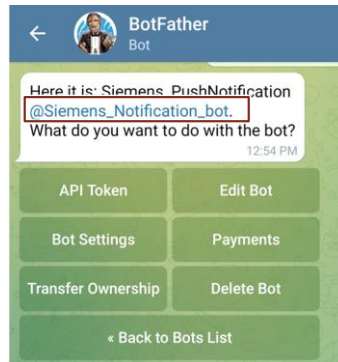


### 3.1.2 Find the chat ID

To send a message to a Telegram chat or a Telegram group, you need the corresponding chat ID of the session.

Proceed as follows to find out what the chat ID is:

1. Click the newly created bot or enter the username of the desired bot in the Telegram search box to start a new conversation with it.



2. Click Start to start a new chat session with the bot.
3. Enter some text and send it to the bot.
4. Launch the URL "https://api.telegram.org/bot[botToken]/getUpdates" in a browser.

#### Note

The [botToken] must be replaced with the respective token (without brackets) provided by the BotFather ([chapter 3.1.1](#)).

#### Result:

getUpdates causes Telegram to return the bot conversation in the following form:

```
{"ok":true,"result":[{"update_id":277706888,"message":{"message_id":3,"from":{"id":3xxxxxxx,"first_name":.....
```

In this example, the chat ID of the Telegram chat session is the '3xxxxxxx'.

#### Note

The new bot can also be added to a group. As in the example above, the chat ID of the group chat will also appear.

The chat ID of a group is negative. It is always preceded by a minus sign.

## 3.2 Alternative messenger services

Other messenger services such as Gotify, Pushbullet or Pushover may also be used for communication between the SIMATIC S7-1500 CPU and the service technician over HTTP (Hypertext Transfer Protocol).

It is not possible to implement the "Receive confirmation from service technician" feature with the messenger services that only support one-sided communication (e.g. Pushover).

The structure of the HTTP requests and the received messages differs between messenger services. Each messenger service uses a different HTTP request method, GET or POST to transmit the requests to the server.

For each messenger service, a separate block must be created, which differs both in terms of the parameters, the format of the requests and the received messages as well as with regard to the internally called library blocks "LHTTP\_Get" or "LHTTP\_PostPut".

The code of the FB "TelegramPushNotification" can be used as a basis for this.

The blocks of the "LHTTP" library, part of the Libraries for Communication for SIMATIC Controllers

(<https://support.industry.siemens.com/cs/ww/en/view/109780503>), are needed to create the blocks.



## 4 Appendix

### 4.1 Service and support

#### Industry Online Support

Do you have any questions or need assistance?

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

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

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

[support.industry.siemens.com](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:

[support.industry.siemens.com/cs/my/src](https://support.industry.siemens.com/cs/my/src)

#### SITRAIN – Digital Industry Academy

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

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

[siemens.com/sitrain](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)

## 4.2 Industry Mall



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

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

## 4.3 Links and literature

Table 4-1

No.	Topic
\1\	Siemens Industry Online Support <a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a>
\2\	Link to the article page of the application example <a href="https://support.industry.siemens.com/cs/ww/en/view/109763879">https://support.industry.siemens.com/cs/ww/en/view/109763879</a>
\3\	Libraries for Communication for SIMATIC Controllers <a href="https://support.industry.siemens.com/cs/ww/en/view/109780503">https://support.industry.siemens.com/cs/ww/en/view/109780503</a>
\4\	SIMATIC S7-1500 CPU 1511-1 PN <a href="https://support.industry.siemens.com/cs/ww/en/view/109752841">https://support.industry.siemens.com/cs/ww/en/view/109752841</a>
\5\	SIMATIC NET: Industrial Remote Communication Remote Networks SCALANCE M-800 Web Based Management <a href="https://support.industry.siemens.com/cs/ww/en/view/109751635">https://support.industry.siemens.com/cs/ww/en/view/109751635</a>
\6\	SIMATIC NET: Industrial Remote Communication - Remote Networks – SCALANCE MUM 856 <a href="https://support.industry.siemens.com/cs/ww/en/view/109802058">https://support.industry.siemens.com/cs/ww/en/view/109802058</a>
\7\	Telegram Bot API - Telegram Methods <a href="https://core.telegram.org/bots/api">https://core.telegram.org/bots/api</a>

## 4.4 Change documentation

Table 4-2

Version	Date	Change
V1.0	04/2022	First edition