<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preface</td>
</tr>
<tr>
<td>2</td>
<td>Documentation guide</td>
</tr>
<tr>
<td>3</td>
<td>Product overview</td>
</tr>
<tr>
<td>4</td>
<td>Communications services</td>
</tr>
<tr>
<td>5</td>
<td>PG communication</td>
</tr>
<tr>
<td>6</td>
<td>HMI communication</td>
</tr>
<tr>
<td>7</td>
<td>Open User Communication</td>
</tr>
<tr>
<td>8</td>
<td>S7 communication</td>
</tr>
<tr>
<td>9</td>
<td>Point-to-point link</td>
</tr>
<tr>
<td>10</td>
<td>OPC UA communication</td>
</tr>
<tr>
<td>11</td>
<td>Routing</td>
</tr>
<tr>
<td>12</td>
<td>Connection resources</td>
</tr>
<tr>
<td>13</td>
<td>Diagnostics and fault correction</td>
</tr>
<tr>
<td>14</td>
<td>Communication with the redundant system S7-1500R/H</td>
</tr>
<tr>
<td></td>
<td>Industrial Ethernet Security with CP 1543-1</td>
</tr>
</tbody>
</table>
Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

**DANGER**
indicates that death or severe personal injury **will** result if proper precautions are not taken.

**WARNING**
indicates that death or severe personal injury **may** result if proper precautions are not taken.

**CAUTION**
indicates that minor personal injury can result if proper precautions are not taken.

**NOTICE**
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

**WARNING**
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.
Preface

Purpose of the documentation

This function manual provides you with an overview of the communication options, the CPUs, communication modules and processors and PC systems of the SIMATIC S7-1500, ET 200MP, ET 200SP, ET 200AL and ET 200pro systems. This function manual describes the connection-oriented, asynchronous communication.

The documentation covers the following:

- Overview of the communication services
- Properties of the communication services
- Overview of the user activities for setting up the communication services

Basic knowledge required

The following knowledge is required in order to understand the Function manual:

- General knowledge of automation technology
- Knowledge of the industrial automation system SIMATIC
- Knowledge about how to use STEP 7 (TIA Portal)

Scope of the documentation

This documentation is the basic documentation for all products of the SIMATIC S7-1500, ET 200MP, ET 200SP, ET 200AL and ET 200pro systems. The product documentation is based on this documentation.

<table>
<thead>
<tr>
<th>What's new?</th>
<th>What are the customer benefits?</th>
<th>Where can I find information?</th>
</tr>
</thead>
<tbody>
<tr>
<td>New contents</td>
<td>Description of communication with the redundant system S7-1500R/H</td>
<td>Section Communication with the redundant system S7-1500R/H [Page 283]</td>
</tr>
<tr>
<td></td>
<td>You receive information on the particularities of communication with the redundant system S7-1500R/H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Functions with which you are familiar from the SIMATIC S7-1500 automation system are implemented for the redundant system S7-1500R/H.</td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>What's new?</th>
<th>What are the customer benefits?</th>
<th>Where can I find the information?</th>
</tr>
</thead>
<tbody>
<tr>
<td>New contents OPC UA Companion Specification</td>
<td>Through OPC UA Companion Specification, methods can be specified in a uniform and manufacturer-neutral way. Using these specified methods, you can easily integrate devices from various manufacturers into the plant and the production processes.</td>
<td>Section AUTOHOTSPOT [Page 283]</td>
</tr>
<tr>
<td>Setting up a secure connection to a mail server over the CPU interface</td>
<td>You can set up a secure connection to a mail server without additional hardware.</td>
<td>Section Secure OUC via e-mail [Page 109]</td>
</tr>
<tr>
<td>Secure communication over Modbus TCP</td>
<td>You can establish secure TCP connections between a Modbus TCP client and a Modbus TCP server.</td>
<td>Section Secure OUC with Modbus TCP [Page 109]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>What's new?</th>
<th>What are the customer benefits?</th>
<th>Where can I find the information?</th>
</tr>
</thead>
<tbody>
<tr>
<td>New contents</td>
<td>OPC UA server</td>
<td>Section AUTOHOTSPOT</td>
</tr>
<tr>
<td></td>
<td>OPC UA is a uniform standard for data communication and is independent of any particular operating system platforms. OPC UA uses integrated safety mechanisms on various automation systems, for example with data exchange, at application level, for the legitimation of the user. The OPC UA server provides a large amount of data: • Values of PLC tags that clients can access • Data types of these PLC tags • Information about the OPC UA server itself and the CPU In this way, clients can gain an overview of the tag management and can read and write values.</td>
<td></td>
</tr>
<tr>
<td>Certificate handling in STEP 7</td>
<td>You can manage certificates for the following applications in STEP 7: • OPC UA server • Secure Open User Communication • Web server of the CPU</td>
<td>Section Managing certificates with STEP 7 (Page 46).</td>
</tr>
<tr>
<td>Deactivating SNMP for the CPU</td>
<td>You can deactivate SNMP for the CPU. This can make sense under certain conditions, for example if the security guidelines in your network do not permit SNMP.</td>
<td>Section Disabling SNMP (Page 60).</td>
</tr>
</tbody>
</table>
Conventions

**STEP 7:** We refer to "STEP 7" in this documentation as a synonym for the configuration and programming software "STEP 7 as of V12 (TIA Portal)".

"S7-1500 CPUs" also refers to the CPU variants S7-1500F, S7-1500T, S7-1500TF, S7-1500C as well as S7-1500pro CPUs, ET200SP CPUs as well as the SIMATIC S7-1500 SW controller.

This documentation contains pictures of the devices described. The figures may differ slightly from the device supplied.

You should also pay particular attention to notes such as the one shown below:

---

**Note**

A note contains important information on the product, on handling of the product and on the section of the documentation to which you should pay particular attention.

---

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](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 customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under [https://www.siemens.com/industrialsecurity](https://www.siemens.com/industrialsecurity).
Siemens Industry Online Support

You can find current information on the following topics quickly and easily here:

- **Product support**
  All the information and extensive know-how on your product, technical specifications, FAQs, certificates, downloads, and manuals.

- **Application examples**
  Tools and examples to solve your automation tasks – as well as function blocks, performance information and videos.

- **Services**
  Information about Industry Services, Field Services, Technical Support, spare parts and training offers.

- **Forums**
  For answers and solutions concerning automation technology.

- **mySupport**
  Your personal working area in Industry Online Support for messages, support queries, and configurable documents.

This information is provided by the Siemens Industry Online Support in the Internet [https://support.industry.siemens.com](https://support.industry.siemens.com).

Industry Mall

The Industry Mall is the catalog and order system of Siemens AG for automation and drive solutions on the basis of Totally Integrated Automation (TIA) and Totally Integrated Power (TIP).

You can find catalogs for all automation and drive products on the Internet [https://mall.industry.siemens.com](https://mall.industry.siemens.com).
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>3</td>
</tr>
<tr>
<td>1 Documentation guide</td>
<td>12</td>
</tr>
<tr>
<td>2 Product overview</td>
<td>17</td>
</tr>
<tr>
<td>3 Communications services</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Overview of communication options</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Communications protocols and port numbers used for Ethernet</td>
<td>25</td>
</tr>
<tr>
<td>3.3 Overview of connection resources</td>
<td>30</td>
</tr>
<tr>
<td>3.4 Setting up a connection</td>
<td>30</td>
</tr>
<tr>
<td>3.5 Data consistency</td>
<td>34</td>
</tr>
<tr>
<td>3.6 Secure Communication</td>
<td>37</td>
</tr>
<tr>
<td>3.6.1 Basics of Secure Communication</td>
<td>37</td>
</tr>
<tr>
<td>3.6.2 Confidentiality through encryption</td>
<td>40</td>
</tr>
<tr>
<td>3.6.3 Authenticity and integrity through signatures</td>
<td>43</td>
</tr>
<tr>
<td>3.6.4 Managing certificates with STEP 7</td>
<td>46</td>
</tr>
<tr>
<td>3.6.5 Examples for the management of certificates</td>
<td>50</td>
</tr>
<tr>
<td>3.6.6 Example: HTTP over TLS</td>
<td>56</td>
</tr>
<tr>
<td>3.7 SNMP</td>
<td>60</td>
</tr>
<tr>
<td>3.7.1 Disabling SNMP</td>
<td>60</td>
</tr>
<tr>
<td>3.7.2 Example: Disabling SNMP for a CPU 1516-3 PN/DP</td>
<td>61</td>
</tr>
<tr>
<td>4 PG communication</td>
<td>63</td>
</tr>
<tr>
<td>5 HMI communication</td>
<td>65</td>
</tr>
<tr>
<td>6 Open User Communication</td>
<td>67</td>
</tr>
<tr>
<td>6.1 Overview of Open User Communication</td>
<td>67</td>
</tr>
<tr>
<td>6.2 Protocols for Open User Communication</td>
<td>68</td>
</tr>
<tr>
<td>6.3 Instructions for Open User Communication</td>
<td>70</td>
</tr>
<tr>
<td>6.4 Open User Communication with addressing via domain names</td>
<td>75</td>
</tr>
<tr>
<td>6.5 Setting up Open User Communication via TCP, ISO-on-TCP, UDP and ISO</td>
<td>77</td>
</tr>
<tr>
<td>6.6 Setting up communication over FDL</td>
<td>83</td>
</tr>
<tr>
<td>6.7 Setting up communication with Modbus TCP</td>
<td>86</td>
</tr>
<tr>
<td>6.8 Setting up communication via e-mail</td>
<td>89</td>
</tr>
<tr>
<td>6.9 Setting up communication via FTP</td>
<td>90</td>
</tr>
<tr>
<td>6.10 Establishment and termination of communications relations</td>
<td>93</td>
</tr>
</tbody>
</table>
Table of contents

6.11 Secure Open User Communication ................................................................. 94
6.11.1 Secure OUC of an S7-1500 CPU as TLS client to an external PLC (TLS server) .......... 94
6.11.2 Secure OUC of an S7-1500 CPU as TLS server to an external PLC (TLS client) .......... 97
6.11.3 Secure OUC between two S7-1500 CPUs ..................................................... 99
6.11.4 Secure OUC via CP interface ......................................................................... 103
6.11.5 Secure OUC with Modbus TCP ..................................................................... 108
6.11.6 Secure OUC via e-mail .................................................................................. 109

7 S7 communication ............................................................................................... 114

8 Point-to-point link ............................................................................................... 123

9 OPC UA communication ...................................................................................... 128

9.1 What you need to know about OPC UA ............................................................. 128
9.1.1 OPC UA and Industrie 4.0 .............................................................................. 128
9.1.2 OPC UA for S7-1500 CPUs .......................................................................... 128
9.1.3 General features of OPC UA ......................................................................... 130
9.1.4 From the Classic OPC interface to OPC UA .................................................... 131
9.1.5 Addressing nodes .......................................................................................... 132
9.1.6 Mapping of data types .................................................................................. 136
9.1.7 What you need to know about OPC UA clients .............................................. 138

9.2 Security at OPC UA .......................................................................................... 143
9.2.1 Security settings ........................................................................................... 143
9.2.2 Certificates pursuant to ITU X.509 ................................................................. 144
9.2.3 Certificates with OPC UA .............................................................................. 147
9.2.4 Creating self-signed certificates ..................................................................... 148
9.2.5 Generating PKI key pairs and certificates yourself ......................................... 149
9.2.6 Secure transfer of messages ......................................................................... 152

9.3 Using the S7-1500 as an OPC UA server ......................................................... 155
9.3.1 Useful information about the S7-1500 CPU OPC UA server ................................ 155
9.3.1.1 The OPC UA server of the S7-1500 CPUs ...................................................... 155
9.3.1.2 End points of the OPC UA server .................................................................. 157
9.3.1.3 Runtime behavior of the OPC UA server ....................................................... 159
9.3.1.4 Diagnostics of the OPC UA server ............................................................... 161
9.3.2 Configuring access to PLC tags ..................................................................... 163
9.3.2.1 Managing write and read rights .................................................................. 163
9.3.2.2 Managing write and read rights for a complete DB ..................................... 165
9.3.2.3 Accessing OPC UA server data ................................................................. 166
9.3.2.4 Export OPC UA XML file ........................................................................... 167
9.3.2.5 Configuring the OPC UA server of the S7-1500 CPU ................................. 168
9.3.3 Enabling the OPC UA server ......................................................................... 168
9.3.3.1 Access to the OPC UA server .................................................................... 170
9.3.3.2 General settings of the OPC UA server ...................................................... 172
9.3.3.3 Settings of the server for subscriptions ....................................................... 173
9.3.3.4 Handling client and server certificates ....................................................... 175
9.3.3.5 Handling of the client certificates of the S7-1500 CPU ............................... 180
9.3.3.6 Generating server certificates with STEP 7 ................................................. 183
9.3.3.7 Editing the security settings of the OPC UA server .................................... 186
9.3.3.8 User authentication .................................................................................. 189
9.3.3.9 Users and roles with OPC UA function rights ........................................... 190
9.3.3.10 Licenses for the OPC UA server .............................................................. 192

9.4 Security at OPC UA .......................................................................................... 193
9.4.1 Security at OPC UA ..................................................................................... 193
9.4.2 Security at OPC UA ..................................................................................... 193
9.4.3 Security at OPC UA ..................................................................................... 193
9.4.4 Security at OPC UA ..................................................................................... 193
# Communication

## Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3.4</td>
<td>Providing methods on the OPC UA server</td>
<td>193</td>
</tr>
<tr>
<td>9.3.4.1</td>
<td>Important facts about server methods</td>
<td>193</td>
</tr>
<tr>
<td>9.3.4.2</td>
<td>Boundary conditions for using server methods</td>
<td>196</td>
</tr>
<tr>
<td>9.3.5</td>
<td>OPC UA server interface configuration</td>
<td>198</td>
</tr>
<tr>
<td>9.3.5.1</td>
<td>What is a server interface?</td>
<td>198</td>
</tr>
<tr>
<td>9.3.5.2</td>
<td>Create a server interface</td>
<td>199</td>
</tr>
<tr>
<td>9.3.5.3</td>
<td>Using OPC UA companion specifications</td>
<td>202</td>
</tr>
<tr>
<td>9.3.5.4</td>
<td>Missing namespaces</td>
<td>217</td>
</tr>
<tr>
<td>9.3.5.5</td>
<td>Coordinating write and read rights for CPU tags</td>
<td>218</td>
</tr>
<tr>
<td>9.3.5.6</td>
<td>Consistency of CPU tags</td>
<td>220</td>
</tr>
<tr>
<td>9.3.5.7</td>
<td>Notes on configuration limits when using server interfaces</td>
<td>222</td>
</tr>
<tr>
<td>9.4</td>
<td>Using the S7-1500 CPU as an OPC UA client</td>
<td>223</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Overview and requirements</td>
<td>223</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Useful information about the client instructions</td>
<td>224</td>
</tr>
<tr>
<td>9.4.3</td>
<td>Number of client instructions that can be used simultaneously</td>
<td>226</td>
</tr>
<tr>
<td>9.4.4</td>
<td>Example configuration for OPC UA</td>
<td>227</td>
</tr>
<tr>
<td>9.4.5</td>
<td>Creating client interfaces</td>
<td>228</td>
</tr>
<tr>
<td>9.4.6</td>
<td>Determine server interface online</td>
<td>237</td>
</tr>
<tr>
<td>9.4.7</td>
<td>Using multilingual texts</td>
<td>241</td>
</tr>
<tr>
<td>9.4.8</td>
<td>Rules for the access to structures</td>
<td>243</td>
</tr>
<tr>
<td>9.4.9</td>
<td>Using connection parameter assignment</td>
<td>245</td>
</tr>
<tr>
<td>9.4.9.1</td>
<td>Creating and configuring connections</td>
<td>245</td>
</tr>
<tr>
<td>9.4.9.2</td>
<td>Handling of the client certificates of the S7-1500 CPU</td>
<td>249</td>
</tr>
<tr>
<td>9.4.9.3</td>
<td>User authentication</td>
<td>252</td>
</tr>
<tr>
<td>9.4.9.4</td>
<td>Using a configured connection</td>
<td>253</td>
</tr>
<tr>
<td>10.1</td>
<td>S7 routing</td>
<td>260</td>
</tr>
<tr>
<td>10.2</td>
<td>Data record routing</td>
<td>265</td>
</tr>
<tr>
<td>11.1</td>
<td>Connection resources of a station</td>
<td>267</td>
</tr>
<tr>
<td>11.2</td>
<td>Allocation of connection resources</td>
<td>271</td>
</tr>
<tr>
<td>11.3</td>
<td>Display of the connection resources</td>
<td>275</td>
</tr>
<tr>
<td>12.1</td>
<td>Connection diagnostics</td>
<td>279</td>
</tr>
<tr>
<td>12.2</td>
<td>Emergency address</td>
<td>282</td>
</tr>
<tr>
<td>13.1</td>
<td>System IP addresses</td>
<td>284</td>
</tr>
<tr>
<td>13.2</td>
<td>Response to Snycup</td>
<td>289</td>
</tr>
<tr>
<td>13.3</td>
<td>Response to primary-backup switchover</td>
<td>289</td>
</tr>
<tr>
<td>13.4</td>
<td>Connection resources of the redundant system S7-1500R/H</td>
<td>290</td>
</tr>
<tr>
<td>13.5</td>
<td>HMI communication with the redundant system S7-1500R/H</td>
<td>292</td>
</tr>
<tr>
<td>13.5.1</td>
<td>HMI connection via the system IP address</td>
<td>292</td>
</tr>
<tr>
<td>13.6</td>
<td>Open User Communication with the redundant system S7-1500R/H</td>
<td>294</td>
</tr>
<tr>
<td>13.6.1</td>
<td>Setting up an Open User Communication connection via the system IP address</td>
<td>295</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>14</td>
<td>Industrial Ethernet Security with CP 1543-1</td>
<td>300</td>
</tr>
<tr>
<td>14.1</td>
<td>Firewall</td>
<td>301</td>
</tr>
<tr>
<td>14.2</td>
<td>Logging</td>
<td>302</td>
</tr>
<tr>
<td>14.3</td>
<td>NTP client</td>
<td>302</td>
</tr>
<tr>
<td>14.4</td>
<td>SNMP</td>
<td>303</td>
</tr>
<tr>
<td>14.5</td>
<td>VPN</td>
<td>303</td>
</tr>
<tr>
<td></td>
<td><strong>Glossary</strong></td>
<td>304</td>
</tr>
<tr>
<td></td>
<td><strong>Index</strong></td>
<td>316</td>
</tr>
</tbody>
</table>
The documentation for the SIMATIC S7-1500 automation system, for CPU 1516pro-2 PN based on SIMATIC S7-1500, and for the distributed I/O systems SIMATIC ET 200MP, ET 200SP and ET 200AL is divided into three areas. This division allows you easier access to the specific information you require.

**Basic information**
System manuals and Getting Started manuals describe in detail the configuration, installation, wiring and commissioning of the SIMATIC S7-1500, ET 200MP, ET 200SP and ET 200AL systems; use the corresponding operating instructions for CPU 1516pro-2 PN. The STEP 7 online help supports you in configuration and programming.

**Device information**
Product manuals contain a compact description of the module-specific information, such as properties, terminal diagrams, characteristics and technical specifications.
General information

The function manuals contain detailed descriptions on general topics such as diagnostics, communication, Motion Control, Web server, OPC UA.

You can download the documentation free of charge from the Internet [https://support.industry.siemens.com/cs/ww/en/view/109742705].

Changes and additions to the manuals are documented in product information sheets.

You will find the product information on the Internet:


Manual Collections

The Manual Collections contain the complete documentation of the systems put together in one file.

You will find the Manual Collections on the Internet:

- S7-1500/ET 200MP [https://support.industry.siemens.com/cs/ww/en/view/86140384]
- ET 200SP [https://support.industry.siemens.com/cs/ww/en/view/84133942]

"mySupport"

With "mySupport", your personal workspace, you make the best out of your Industry Online Support.

In "mySupport", you can save filters, favorites and tags, request CAx data and compile your personal library in the Documentation area. In addition, your data is already filled out in support requests and you can get an overview of your current requests at any time.

You must register once to use the full functionality of "mySupport".

You can find "mySupport" on the Internet [https://support.industry.siemens.com/My/ww/en].

"mySupport" - Documentation

In the Documentation area in "mySupport" you can combine entire manuals or only parts of these to your own manual.

You can export the manual as PDF file or in a format that can be edited later.

You can find "mySupport" - Documentation on the Internet [https://support.industry.siemens.com/My/ww/en/documentation].
"mySupport" - CAx data

In the CAx data area in "mySupport", you can access the current product data for your CAx or CAe system.

You configure your own download package with a few clicks.

In doing so you can select:

- Product images, 2D dimension drawings, 3D models, internal circuit diagrams, EPLAN macro files
- Manuals, characteristics, operating manuals, certificates
- Product master data

You can find "mySupport" - CAx data on the Internet [https://support.industry.siemens.com/my/ww/en/CAxOnline].

Application examples

The application examples support you with various tools and examples for solving your automation tasks. Solutions are shown in interplay with multiple components in the system - separated from the focus on individual products.

You will find the application examples on the Internet [https://support.industry.siemens.com/sc/ww/en/sc/2054].

TIA Selection Tool

With the TIA Selection Tool, you can select, configure and order devices for Totally Integrated Automation (TIA).

This tool is the successor of the SIMATIC Selection Tool and combines the known configurators for automation technology into one tool.

With the TIA Selection Tool, you can generate a complete order list from your product selection or product configuration.

You can find the TIA Selection Tool on the Internet [https://w3.siemens.com/mcms/topics/en/simatic/tia-selection-tool].
SIMATIC Automation Tool

You can use the SIMATIC Automation Tool to run commissioning and maintenance activities simultaneously on different SIMATIC S7 stations as a bulk operation, independently of the TIA Portal.

The SIMATIC automation tool provides a variety of functions:

- Scanning of a PROFINET/Ethernet plant network and identification of all connected CPUs
- Address assignment (IP, subnet, gateway) and station name (PROFINET device) to a CPU
- Transfer of the date and programming device/PC time converted to UTC time to the module
- Program download to CPU
- Operating mode switchover RUN/STOP
- CPU localization by means of LED flashing
- Reading out CPU error information
- Reading of CPU diagnostic buffer
- Reset to factory settings
- Updating the firmware of the CPU and connected modules


PRONETA

With SIEMENS PRONETA (PROFINET network analysis), you analyze the plant network during commissioning. PRONETA features two core functions:

- The topology overview independently scans PROFINET and all connected components.
- The IO check is a fast test of the wiring and the module configuration of a plant.

SINETPLAN

SINETPLAN, the Siemens Network Planner, supports you in planning automation systems and networks based on PROFINET. The tool facilitates professional and predictive dimensioning of your PROFINET installation as early as in the planning stage. In addition, SINETPLAN supports you during network optimization and helps you to exploit network resources optimally and to plan reserves. This helps to prevent problems in commissioning or failures during productive operation even in advance of a planned operation. This increases the availability of the production plant and helps improve operational safety.

The advantages at a glance

● Network optimization thanks to port-specific calculation of the network load
● Increased production availability thanks to online scan and verification of existing systems
● Transparency before commissioning through importing and simulation of existing STEP 7 projects
● Efficiency through securing existing investments in the long term and optimal exploitation of resources

You can find SINETPLAN on the Internet [https://www.siemens.com/sinetplan](https://www.siemens.com/sinetplan).
Product overview

CPUs, communications modules and processors, and PC systems of the S7-1500, ET 200MP, ET 200SP, ET 200pro and ET 200AL systems provide you with interfaces for communication via PROFINET, PROFIBUS and point-to-point connections.

CPUs, communications modules and communications processors

PROFINET and PROFIBUS DP interfaces are integrated in the S7-1500 CPUs. The CPU 1516-3 PN/DP for example has two PROFINET interfaces and one PROFIBUS DP interface. Other PROFINET and PROFIBUS DP interfaces are available by using communications modules (CM) and communications processors (CP).

![Diagram of CPU 1516-3 PN/DP and CPU 1512SP-1 PN]

1. PROFINET interface (X2) with 1 port
2. PROFINET interface (X1) with 2-port switch
3. PROFIBUS DP interface (X3)
4. PROFINET interface (X1) with 3-port switch

Figure 2-1 Interfaces of the CPU 1516-3 PN/DP and CPU 1512SP-1 PN
Interfaces of communications modules

Interfaces of communications modules (CMs) extend the interfaces of CPUs (for example, the communication module CM 1542-5 adds a PROFIBUS interface to S7-1500 automation system).

PROFIBUS DP interface

Figure 2-2 PROFIBUS DP interface of the CM 1542-5 and CM DP
Interfaces of communications processors

Interfaces of communications processors (CPs) provide an additional functionality compared with the integrated interfaces of the CPUs. CPs allow special applications, for example the CP 1543-1 provides Industrial Ethernet security functions for protecting Industrial Ethernet networks via its Industrial Ethernet interface.

![Industrial Ethernet interface of the CP 1543-1](image)

1 Industrial Ethernet interface

Figure 2-3 Industrial Ethernet interface of the CP 1543-1
Interfaces of communications modules for point-to-point connections

The communication modules for point-to-point connections provide communication via their RS 232-, RS 422- and RS 485 interfaces, for example, Freeport or Modbus communication.

Figure 2-4 Example of interface for point-to-point connection at the CM PIP RS422/485 BA
Interfaces of Interface modules

PROFINET and PROFIBUS DP interfaces of the interface modules (IM) in ET 200MP, ET 200SP and ET 200AL are used to connect the distributed I/O ET 200MP, ET 200SP and ET 200AL to PROFINET or PROFIBUS of the higher-level IO controller or DP master.

![Image of IM 155-5 PN ST, IM 155-6 PN ST, and IM 157-1 PN]

PROFINET interface with 2-port switch

Figure 2-5   PROFINET interfaces IM 155-5 PN ST (ET 200MP), IM 155-6 PN ST (ET 200SP), and IM 157-1 PN (ET 200AL)

Communications services

The communications services described below use the interfaces and communication mechanisms provided by the system via CPUs, communication modules and processors.
# Communications services

## 3.1 Overview of communication options

### Overview of communications options

The following communications options are available for your automation task.

Table 3-1 Communications options

<table>
<thead>
<tr>
<th>Communications options</th>
<th>Functionality</th>
<th>Via interface:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG communication&lt;sup&gt;2&lt;/sup&gt;</td>
<td>On commissioning, testing, diagnostics</td>
<td>X X -</td>
</tr>
<tr>
<td>HMI communication&lt;sup&gt;2&lt;/sup&gt;</td>
<td>On operator control and monitoring</td>
<td>X X -</td>
</tr>
<tr>
<td>Open communication via TCP/IP&lt;sup&gt;2&lt;/sup&gt;</td>
<td>Data exchange via PROFINET/Industrial Ethernet with TCP/IP</td>
<td>X - -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND/TRCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TCON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• T_DISCON</td>
<td></td>
</tr>
<tr>
<td>Open communication using ISO-on-TCP&lt;sup&gt;2&lt;/sup&gt;</td>
<td>Data exchange via PROFINET/Industrial Ethernet with ISO-on-TCP</td>
<td>X - -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND/TRCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TCON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• T_DISCON</td>
<td></td>
</tr>
<tr>
<td>Open communication via UDP&lt;sup&gt;2&lt;/sup&gt;</td>
<td>Data exchange via PROFINET/Industrial Ethernet with UDP</td>
<td>X - -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TUSEND/TRUCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TCON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• T_DISCON</td>
<td></td>
</tr>
</tbody>
</table>
### 3.1 Overview of communication options

<table>
<thead>
<tr>
<th>Communications options</th>
<th>Functionality</th>
<th>Via interface:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open communication via ISO (only CPs with PROFINET/Industrial Ethernet interface)</td>
<td>Data exchange via PROFINET/Industrial Ethernet with the ISO protocol</td>
<td>PN/IE1: X, DP: - , serial: -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND/TRCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TCON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• T_DISCON</td>
<td></td>
</tr>
<tr>
<td>Open communication with FDL (only CM 1542-5 as of firmware V2.0)</td>
<td>Data exchange via PROFIBUS with the FDL protocol</td>
<td>PN/IE1: - , DP: X, serial: -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TSEND/TRCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TUSEND/TURCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TCON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• T_DISCON</td>
<td></td>
</tr>
<tr>
<td>OPC UA server (only via internal PROFINET interfaces of the CPU)</td>
<td>Data exchange with OPC UA clients</td>
<td>PN/IE1: X, DP: - , serial: -</td>
</tr>
<tr>
<td>Communication via Modbus TCP</td>
<td>Data exchange via PROFINET with Modbus TCP protocol</td>
<td>PN/IE1: X, DP: - , serial: -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MB_CLIENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MB_SERVER</td>
<td></td>
</tr>
<tr>
<td>E-mail</td>
<td>Sending process alarms via e-mail</td>
<td>PN/IE1: X, DP: - , serial: -</td>
</tr>
<tr>
<td></td>
<td>Instruction:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TMAIL_C</td>
<td></td>
</tr>
<tr>
<td>FTP (only CPs with PROFINET/Industrial Ethernet interface)</td>
<td>File management and file access via FTP (File Transfer Protocol); CP can be FTP client and FTP server</td>
<td>PN/IE1: X, DP: - , serial: -</td>
</tr>
<tr>
<td></td>
<td>Instruction:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• FTP_CMD</td>
<td></td>
</tr>
<tr>
<td>Fetch/Write (only CPs with PROFINET/Industrial Ethernet interface)</td>
<td>Server services via TCP/IP, ISO-on-TCP and ISO</td>
<td>PN/IE1: X, DP: - , serial: -</td>
</tr>
<tr>
<td></td>
<td>Via special instructions for Fetch/Write</td>
<td></td>
</tr>
<tr>
<td>S7 communication</td>
<td>Data exchange via PROFINET/PROFIBUS with the S7 protocol.</td>
<td>PN/IE1: X, DP: X, serial: -</td>
</tr>
<tr>
<td></td>
<td>Instructions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PUT/GET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• BSEND/BRCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• USEND/URCV</td>
<td></td>
</tr>
</tbody>
</table>
## 3.1 Overview of communication options

<table>
<thead>
<tr>
<th>Communications options</th>
<th>Functionality</th>
<th>Via interface:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial point-to-point connection</td>
<td>Data exchange via point-to-point with Freeport, 3964(R), USS or Modbus protocol</td>
<td>PN/IE, DP, serial</td>
</tr>
<tr>
<td></td>
<td>Via special instructions for PtP, USS or Modbus RTU</td>
<td></td>
</tr>
<tr>
<td>Web server</td>
<td>Data exchange via HTTP(S), for example for diagnostics</td>
<td>PN/IE, DP, serial</td>
</tr>
<tr>
<td>SNMP (Simple Network Management Protocol)</td>
<td>For monitoring and error recognition of IP networks, possibly parameterization of the IP network components via standard SNMP protocol</td>
<td>PN/IE, DP, serial</td>
</tr>
<tr>
<td>Time-of-day synchronization</td>
<td>Via PN/IE interface: CPU is NTP client (Network Time Protocol)</td>
<td>PN/IE, DP, serial</td>
</tr>
<tr>
<td></td>
<td>Via DP interface: CPU/CM/CP is time-of-day master or time slave</td>
<td>PN/IE, DP, serial</td>
</tr>
</tbody>
</table>

1. IE - Industrial Ethernet
2. Observe the special characteristics for S7-1500R/H

### Information on S7-1500R/H

You can find information on the communication possibilities with the redundant system S7-1500R/H in the section Communication with the redundant system S7-1500R/H (Page 283).

### Additional information

- Additional information about the Fetch/Write services is available in the STEP 7 online help.
3.2 Communications protocols and port numbers used for Ethernet communication

This section provides an overview of the supported protocols and port numbers used for communication over PN/IE interfaces. For each protocol the address parameters, the respective communications layer as well as the communications role and the communications direction are specified.

This information makes it possible to match the security measures for protection of the automation system to the used protocols (for example firewall). Because security measures are limited to Ethernet or PROFINET networks, the tables do not include PROFIBUS protocols.

Note
Port numbers used

The specified port numbers are the standard port numbers used by the S7-1500 CPU. Many communication protocols and implementations enable you to use other port numbers.

The tables below show the different layers and protocols that are being used.

The following table shows the protocols supported by the S7-1500 CPUs, ET 200SP CPUs and the CPU 1516pro-2 PN. The S7-1500 software controllers also support the protocols listed in the following table for the Ethernet interfaces that are assigned to the software controller.

Table 3-2 Layers and protocols of the S7-1500 CPUs and software controllers (via PROFINET interface of the CPU)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(2) Link layer (4) Transport layer</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFINET protocols</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCP Discovery and basic config-</td>
<td>Not relevant</td>
<td>(2) Ethertype 0x8892 (PROFINET)</td>
<td>Accessible devices</td>
<td>DCP is used by PROFINET to discover PROFINET devices and provide basic settings.</td>
</tr>
<tr>
<td>uration protocol</td>
<td></td>
<td></td>
<td>PROFINET Discovery and configuration</td>
<td></td>
</tr>
<tr>
<td>LLDP Link Layer Discovery</td>
<td>Not relevant</td>
<td>(2) Ethertype 0x88CC (LLDP)</td>
<td>PROFINET Link Layer Discovery protocol</td>
<td>LLDP is used by PROFINET to discover and manage neighbor relationships between PROFINET devices. LLDP uses the special multicast MAC address: 01-80-C2-00-00-0E</td>
</tr>
<tr>
<td>protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRP Media Redundancy Protocol</td>
<td>Not relevant</td>
<td>(2) Ethertype 0x88F3 (IEC 62493-2-2010)</td>
<td>PROFINET medium redundancy</td>
<td>MRP provides control of redundant transmission paths by means of a ring topology. MRP uses standard-compliant Multicast MAC addresses.</td>
</tr>
<tr>
<td>PTCP Precision Transparent</td>
<td>Not relevant</td>
<td>(2) Ethertype 0x8892 (PROFINET)</td>
<td>PROFINET send clock and time synchronization, based on IEEE 1588</td>
<td>PTC provides a time delay measurement between RJ45 ports and thus send clock synchronization and time synchronization. PTCP uses standard-compliant Multicast MAC addresses.</td>
</tr>
<tr>
<td>Clock Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Communications services

### 3.2 Communications protocols and port numbers used for Ethernet communication

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(2) Link layer (4) Transport layer</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFINET IO data</td>
<td>Not relevant</td>
<td>(2) Ethertype 0x8892 (PROFINET)</td>
<td>PROFINET Cyclic IO data transfer</td>
<td>The PROFINET IO frames are used to transmit IO data cyclically between PROFINET IO controller and IO devices via Ethernet.</td>
</tr>
<tr>
<td>PROFINET Context Manager</td>
<td>34964</td>
<td>(4) UDP</td>
<td>PROFINET connection less RPC</td>
<td>The PROFINET Context Manager provides an endpoint mapper in order to establish an application relation (PROFINET AR).</td>
</tr>
</tbody>
</table>

#### Connection-oriented communications protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(4) TCP</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP</td>
<td>25</td>
<td></td>
<td>Simple mail transfer protocol</td>
<td>SMTP is used for sending e-mails.</td>
</tr>
<tr>
<td>SMTPS (SMTP over TLS)</td>
<td>465</td>
<td></td>
<td>Secure SMTP</td>
<td>SMTP is used for sending e-mails over secure connections.</td>
</tr>
<tr>
<td>SMTP with STARTTLS</td>
<td>25</td>
<td>587</td>
<td>Simple mail transfer protocol with the SMTP command &quot;STARTTLS&quot;</td>
<td>SMTP with STARTTLS is used for sending e-mails over secure connections.</td>
</tr>
<tr>
<td>HTTP</td>
<td>80</td>
<td></td>
<td>Hypertext transfer protocol</td>
<td>HTTP is used for communication with the CPU-internal web server.</td>
</tr>
<tr>
<td>ISO-on-TCP (according to RFC 1006)</td>
<td>102</td>
<td></td>
<td>ISO-on-TCP protocol</td>
<td>ISO-on-TCP (according to RFC 1006) is used for message-oriented data exchange with remote CPU or software controller.</td>
</tr>
<tr>
<td>NTP</td>
<td>123</td>
<td></td>
<td>Network time protocol</td>
<td>NTP is used for synchronization of the CPU system time with the time of an NTP server.</td>
</tr>
<tr>
<td>SNMP</td>
<td>161</td>
<td>162 (trap)</td>
<td>Simple network management protocol</td>
<td>SNMP is used for reading and setting of network management data (SNMP managed Objects) by the SNMP Manager.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>443</td>
<td></td>
<td>Secure Hypertext transfer protocol</td>
<td>HTTPS is used for communication with the CPU-internal web server via Secure Socket Layer (SSL).</td>
</tr>
<tr>
<td>Modbus TCP Modbus Transmission Control Protocol</td>
<td>502</td>
<td></td>
<td>Modbus/TCP protocol</td>
<td>Modbus/TCP is used by MB_CLIENT/MB_SERVER instructions in the user program.</td>
</tr>
</tbody>
</table>
## 3.2 Communications protocols and port numbers used for Ethernet communication

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(2) Link layer (4) Transport layer</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC UA</td>
<td>4840</td>
<td>(4) TCP</td>
<td>Based on the TCP/IP protocol</td>
<td>Communication standard ranging from the enterprise level to the field level.</td>
</tr>
<tr>
<td>Open Platform Communications Unified Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUC¹</td>
<td>1 ... 1999</td>
<td>(4) TCP (4) UDP</td>
<td>Open User Communication (TCP/UDP) Secure Open User Communication (TLS)</td>
<td>OUC instructions provide connection establishment, connection termination and data transfer based on the socket layer.</td>
</tr>
<tr>
<td>Open User Communication and Secure OUC</td>
<td>2000 ... 5000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>49152 ... 65535</td>
<td>(4) TCP (4) UDP</td>
<td></td>
<td>Dynamic port area used for active connection endpoint if the application does not determine the local port number.</td>
</tr>
<tr>
<td>IGMPv2</td>
<td>Not relevant</td>
<td>(3) Network layer</td>
<td>Internet Group Management Protocol</td>
<td>Network protocol for the organization of multicast groups.</td>
</tr>
<tr>
<td>Internet Group Management Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹ Note: The open communication provides direct access to the UDP/TCP for the user. The user is responsible for observing the port restrictions/definitions of the IANA (Internet Assigned Numbers Authority).

² Do not use ports for OUC, which are already used by other protocols.

The following table shows the protocols that are supported by the S7-1500 software controller via the Ethernet interfaces assigned to Windows.

### Table 3- 3 Layers and protocols of the S7-1500 Software Controller (via Ethernet interface on the Windows side)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(2) Link layer (4) Transport layer</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFINET protocols</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCP</td>
<td>Not relevant</td>
<td>(2) Ethertype 0x8892 (PROFINET)</td>
<td>Accessible devices PROFINET Discovery and configuration</td>
<td>DCP is used by PROFINET to discover PROFINET devices and provide basic settings.</td>
</tr>
<tr>
<td>Discovery and basic configuration protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 3.2 Communications protocols and port numbers used for Ethernet communication

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(2) Link layer</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP</td>
<td>25</td>
<td>(4) TCP</td>
<td>Simple mail transfer protocol</td>
<td>SMTP is used for sending e-mails.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Adjustable 1</td>
<td>(4) TCP</td>
<td>Hypertext transfer protocol</td>
<td>HTTP is used for communication with CPU-internal web server. You can change the port number to avoid conflict with other web servers on Windows. If you want to use web server access, you must activate the port in the Windows Firewall.</td>
</tr>
<tr>
<td>ISO-on-TCP</td>
<td>102</td>
<td>(4) TCP</td>
<td>ISO-on-TCP protocol</td>
<td>ISO-on-TCP (according to RFC 1006) for S7 communication with PG/PC or HMI.</td>
</tr>
<tr>
<td>OUC2</td>
<td>1 ... 1999 can be used to limited extent</td>
<td>(4) TCP (4) UDP</td>
<td>Open User Communication (TCP/UDP) Secure Open User Communication (TLS)</td>
<td>OUC instructions provide connection establishment, connection termination and data transfer based on the socket layer. If you want to use OUC, you must activate the ports in the Windows Firewall.</td>
</tr>
<tr>
<td>IGMPv2</td>
<td>Not relevant</td>
<td>(3) Network layer</td>
<td>Internet Group Management Protocol</td>
<td>Network protocol for the organization of multicast groups.</td>
</tr>
<tr>
<td>Reserved</td>
<td>49152 ... 65535</td>
<td>(4) TCP (4) UDP</td>
<td>-</td>
<td>Dynamic port range that is used for the active connection end point, if the application does not determine the local port number. If you wish to use this communication, you must activate the ports in the Windows Firewall.</td>
</tr>
</tbody>
</table>

1. Default setting for Windows assigned interfaces: 81
2. Note: The open user communication provides direct access to the UDP/TCP for the user. The user is responsible for observing the port restrictions/definitions of the IANA (Internet Assigned Numbers Authority).
3. Do not use ports for OUC, which are already used by other protocols.
4. Do not use ports for OUC, which are already used by other Windows applications.
The following table shows the protocols that are supported in addition to those listed in the tables for the S7-1500 communications modules (e.g. CP 1543-1).

Table 3-4  Layers and protocols of S7-1500 communications modules

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>(2) Link layer</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFInET/Industrial Ethernet protocols</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTP</td>
<td>20 (data)</td>
<td>(4) TCP</td>
<td>File transfer protocol</td>
<td>FTP is used for the transmission of files (only in connection with CP 1543-1).</td>
</tr>
<tr>
<td>File transfer protocol</td>
<td>21 (control)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>secureFTP</td>
<td>20 (data)</td>
<td>(4) TCP</td>
<td>File transfer protocol</td>
<td>SecureFTP is used for the transmission of files by means of a TSL connection (only in connection with CP 1543-1).</td>
</tr>
<tr>
<td>File transfer protocol</td>
<td>21 (control)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHCP</td>
<td>68</td>
<td>(4) UDP</td>
<td>Dynamic Host Configuration Protocol</td>
<td>DHCP is used to retrieve the IP Address Suite from a DHCP server when starting up the IE interface.</td>
</tr>
<tr>
<td>Dynamic Host Configuration Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure NTPv3</td>
<td>123</td>
<td>(4) UDP</td>
<td>Network time protocol</td>
<td>Secure NTP is used to synchronize the CP 1543-1 internal system clock with an NTP server.</td>
</tr>
<tr>
<td>Network time protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNMP</td>
<td>161</td>
<td>(4) UDP</td>
<td>Simple network management protocol</td>
<td>SNMPv3 permits the CP 1543-1 to read network management data (MIBs) from SNMPv3 agent with authentication.</td>
</tr>
<tr>
<td>Simple network management protocol</td>
<td>162 (trap)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Special consideration S7-1500 MFP:
Port 111: The S7-1500 MFP uses Port 111 to the NFS service for internal communication.
3.3 Overview of connection resources

Connection resources

Some communications services require connections. Connections allocate resources on the CPUs, CPs and CMs involved (for example memory areas in the CPU operating system). In most cases one resource per CPU/CP/CM is allocated for a connection. In HMI communication, up to 3 connection resources are required per HMI connection.

The connection resources available depend on the CPU being used, the CPs and CMs and must not exceed a defined high limit for the automation system.

Available connection resources in a station

The maximum number of resources of a station is determined by the CPU.

Each CPU has reserved connection resources for PG, HMI and web server communication. There are also resources available that can be used for SNMP, e-mail connections, HMI, S7 communication as well as for open communication.

When are connection resources allocated?

The time for allocation of connection resources depends on how the connection is set up, automatic, programmed or configured (see section Setting up a connection (Page 30)).

Additional information

You will find more detailed information on the allocation of connection resources and the display of connection resources in STEP 7 in the section Connection resources (Page 267).

3.4 Setting up a connection

Automatic connection

STEP 7 sets up a connection automatically (for example PG or HMI connection) if you have connected the PG/PC interface to an interface of the CPU physically and have made the interface assignment in STEP 7 in the "Go online" dialog.
Setting up a programmed connection

You set up the programmed connection in the program editor of STEP 7 in the context of a CPU by assigning instructions for communication, for example TSEND_C.

When specifying the connection parameters (in the Inspector window, in the properties of the instruction), you are supported by the easy-to-use user interface.
Setting up a configured connection

You set up the configured connection in the network view of the Devices & networks editor of STEP 7 in the context of a CPU or a software controller.

![Configured setup](image)

Figure 3-2 Configured setup
Effects on the connection resources of the CPU

You can often choose between a configured or a programmed connection. Programmed connection setup allows connection resources to be released following data transfer. Like routed connections, programmed connections are not guaranteed, meaning that they are only established when resources are available. With configured connection setup, the resource is available after download of the configuration until the configuration changes again. Corresponding resources are therefore reserved for connection establishment via configured connections. The "Connection resources" table in the Inspector window of the CPU displays an overview of connection resources already used and those still available.

How do I set up a connection?

Table 3-5 Setting up the connection

<table>
<thead>
<tr>
<th>Connection</th>
<th>Automatically</th>
<th>Programmed setup</th>
<th>Configured setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming device connection</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HMI connection</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Open communication via TCP/IP</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open communication via ISO-on-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TCP connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open communication via UDP</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open communication via ISO</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open communication via FDL</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication via Modbus TCP</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-mail connection</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>FTP connection</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>S7 connection*</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

* Note that for an S7-1500 CPU with firmware version lower than V2.0, the use of PUT/GET communication must be enabled in the properties of the CPU. You can find more information on this topic in the STEP 7 online help.

Additional information

You will find further information on the allocation of connection resources and the display of connection resources in STEP 7 in the section Connection resources (Page 267).
3.5 Data consistency

Definition

Data consistency is important for data transfer and you need to take this into account when configuring the communication task. Otherwise, malfunctions may occur.

A data area which cannot be modified by concurrent processes is called a consistent data area. This means that a data area which belongs together and which is larger than the maximum size of the consistent data area can consist in part of new and of old data at the same time.

An inconsistency can occur when an instruction for communication is interrupted, for example by a hardware interrupt OB with higher priority. This interrupts the transfer of the data area. If the user program in this OB now changes the data that has not yet been processed by the communication instruction, the transferred data originates from different times:

The following figure shows a data area that is smaller than the maximum size of the consistent data area. In this case, when transferring the data area, it is ensured that there is no interruption by the user program during data access so that the data is not changed.

![Consistent transfer of data](image)

1. The source data area is smaller than the maximum size of the consistent data area (2). The instruction transfers the data together to the destination data area.
2. Maximum size of the consistent data area

Figure 3-3 Consistent transfer of data
The following figure shows a data area that is larger than the maximum size of the consistent data area. In this case, the data can be changed during an interruption of the data transfer. An interruption also occurs if, for example, the data area needs to be transferred in several parts. If the data is changed during the interruption, the transferred data originates from different times.

1. The source data area is larger than the maximum size of the consistent data area. At time T1, the instruction only transfers as much data from the source data area into the destination data area as fits in the consistent data area.

2. At time T2, the instruction transfers the rest of the source data area to the destination data area. After the transfer, data from different points in time exist in the destination data area. If the data in the source data area has changed in the meantime, an inconsistency may result.

3. Maximum size of the consistent data area

Example of an inconsistency

The figure below shows an example of changing data during the transfer. The destination data area contains data from different points in time.

1. Maximum size of the consistent data area

Figure 3-4 Transfer of data larger than the maximum consistency area

Figure 3-5 Example: Changing data during the transfer
System-specific maximum data consistency for S7-1500:

No inconsistency occurs if the system-specific maximum size of the consistent data is kept to. With an S7-1500, communication data is copied consistently into or out of the user memory in blocks of up to 512 bytes during the program cycle. Data consistency is not ensured for larger data areas. Where defined data consistency is required, the length of communication data in the user program of the CPU must not exceed 512 bytes. You can then access these data areas consistently, for example from an HMI device by reading/writing tags.

If more data than the system-specific maximum size needs to be transferred consistently, you yourself must ensure the data consistency with suitable measures in the user program.

Ensuring data consistency

Use of instructions for access to common data:

If the user program contains instructions for communication that access common data, for example TSEND/TRCV, you can coordinate access to this data area yourself, for example using the "DONE" parameter. The data consistency of the data areas that are transferred with an instruction for communication can therefore be ensured in the user program.

Note

Measures in the user program

To achieve data consistency, you can copy transferred data to a separate data area (for example, global data block). While the user program continues to work with the original data, you can transfer the data saved in the separate data area consistently with the communication instruction.

For the copying, use uninterruptible instructions such as UMOVE_BLK or UFILL_BLK. These instructions ensure data consistency up to 16 KB.

Use of PUT/GET instructions or Write/Read via HMI communication:

In S7 communication with the PUT/GET instructions or Write/Read via HMI communication, you need to take into account the size of the consistent data areas during programming or configuration. In the user program of an S7-1500 as server, there is no instruction available that can coordinate the data transfer in the user program. The data exchanged using PUT/GET instructions updates the S7-1500 while the user program is running. There is no point in time within the processing of the cyclic user program at which the data is exchanged consistently. The length of the data area to be transferred should be smaller than 512 bytes.

Additional information

- You will also find the maximum amount of consistent data in the device manuals of the communications modules in the Technical Specifications.
- You will find further information on data consistency in the description of the instructions in the STEP 7 online help.
3.6 Secure Communication

3.6.1 Basics of Secure Communication

For STEP 7 (TIA Portal) as of V14 and for S7-1500 CPUs as of firmware V2.0, the options for secure communication have been broadened considerably.

Introduction

The attribute "secure" is used for the identification of communication mechanisms that are based on a Public Key Infrastructure (PKI) (for example RFC 5280 for Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List Profile). A Public Key Infrastructure (PKI) is a system that can issue, distribute and check digital certificates. The digital certificates issued are used in the PKI to secure computer-based communication. If a PKI uses asymmetric key cryptography, the messages in a network can be digitally signed and encrypted.

Components that you have configured in STEP 7 (TIA Portal) for secure communication use an asymmetric key encryption scheme with a Public Key and Private Key. TLS (Transport Layer Security) is used as the encryption protocol. TLS is the successor for the SSL (Secure Sockets Layer) protocol.

Objectives of secure communication

Secure communication is used to achieve the following objectives:

- Confidentiality
  i.e. the data are secret / cannot be read by eavesdroppers.

- Integrity
  i.e. the message that reaches the recipient is the same message, unchanged, that the sender sent. The message has not been altered on the way.

- End point authentication
  i.e. the end point communication partner is exactly who it claims to be and the party who is to be reached. The identity of the partner has been checked.

These objectives were in the past primarily relevant to IT and networked computers. Now, industrial machinery and control systems with sensitive data are at equally high risk, as they are also networked, and consequently pose strict security requirements for data exchange.

Protection of the automation cell by means of the cell protection concept through firewall, or via connection through VPN, for example with the security module, was common in the past and remains so.

However, it is becoming increasingly necessary to also transfer data to external computers in encrypted form via Intranet or public networks.
Common principles of secure communication

Independent of the context, secure communication is based on the concept of the Public Key Infrastructure (PKI) and contains the following components:

- An asymmetric encryption scheme that allows:
  - Encryption or decryption of messages using public or private keys.
  - The verification of signatures in messages and certificates.

  The messages/certificates are signed by the sender/certificate subject with their private key. The recipient/verifier checks the signature with the public key of the sender/certificate subject.

- Transport and storage of the public key using X.509 certificates:
  - X.509 certificates are digitally signed data that allow public key authentication in terms of the bound identity.
  - X.509 certificates can contain information that describes in more detail or restricts use of the public key. For example the date as of which a public key in a certificate is valid and when it expires.
  - X.509 certificates contain information about the issuer of the certificate in secure form.

The following paragraphs give an overview of these basic concepts, which are required for managing certificates in STEP 7 (TIA Portal), for example, and for programming communication instructions for secure Open User Communication (sOUC).
Secure communication with STEP 7

STEP 7 as of V14 provides the required PKI for the configuration and operation of secure communication.

Examples:

- The Hypertext Transfer Protokoll (HTTP) turns into Hypertext Transfer Protokoll Secure (HTTPS) with the help of the TLS (Transport Layer Security) protocol. Since HTTPS is a combination of HTTP and TLS, it is called "HTTP over TLS" in the corresponding RFC. You can see in the browser that HTTPS is being used; this is indicated by the URL "https://" instead of "http://" in the address bar of the browser. Most browsers highlight such secure connections.

- Open User Communication turns into secure Open User Communication. The underlying protocol is also TLS.

- E-mail providers also offer access over the "Secure SMTP over TLS" protocol to increase the security of e-mail communication.

The figure below shows the TLS protocol in the context of communication layers.

Secure communication with OPC UA

An OPC UA server is implemented in S7-1500 CPUs as of firmware V2.0. OPC UA Security also covers authentication, encryption and data integrity with digital X.509 certificates and also uses a Public Key Infrastructure (PKI). Depending on the requirements placed by the application, you can select different security levels for the end point security. You will find the description of the OPC UA server functionality in the section AUTOHOTSPOT.
3.6 Secure Communication

3.6.2 Confidentiality through encryption

Message encryption is an important element of data security. When encrypted messages are intercepted by third parties during communication, these potential eavesdroppers cannot access the information they contain.

There is a wide range of mathematical processes (algorithms) for encrypting messages. All algorithms process a "key" parameter to encrypt and decrypt messages.

- Algorithm + key + message => encrypted message
- Encrypted message + key + algorithm => (decrypted) message

Symmetric encryption

The central aspect of symmetric encryption is that both communication partners use the same key for message encryption and decryption, as shown in the figure below. Bob uses the same key for encryption as Alice uses for decryption. In general, we also say that the two sides share the secret key with which they encrypt or decrypt a message as a secret.

![Symmetric encryption diagram](image)

① Bob encrypts his message with the symmetric key
② Alice decrypts the encrypted message with the symmetric key

Figure 3-7 Symmetric encryption

The process can be compared to a briefcase to which the sender and recipient have the same key, which both locks and opens the case.

- Advantage: Symmetric encryption algorithms (such as AES, Advanced Encryption Algorithm) are fast.
- Disadvantages: How can the key be sent to a recipient without getting into the wrong hands? This is a key distribution problem. If enough messages are intercepted, the key can also be worked out and must therefore be changed regularly.

If there are a large number of communication partners, there is also a large number of keys to distribute.
Asymmetric encryption

Asymmetric encryption works with a pair of keys consisting of one public key and one private key. Used with a PKI, it is also known as Public Key cryptography or simply PKI cryptography. A communication partner, Alice in the figure below, has a private key and a public key. The public key is provided to the public, in other words any potential communication partner. Anyone with the public key can encrypt messages for Alice. In the figure below, this is Bob.

Alice’s private key, which she must not disclose, is used by Alice to decrypt an encrypted message addressed to her.

1. Alice provides Bob with her public key. No precautionary measures are required to this purpose: Anyone can use the public key for messages to Alice if they are sure that it is actually Alice’s public key.

2. Bob encrypts his message with Alice’s public key.

3. Alice decrypts the encrypted message from Bob with her private key. As only Alice has the private key and never discloses it, only she can decrypt the message. With her private key, she can decrypt any message encrypted with her public key - not only messages from Bob.

Figure 3-8 Asymmetric encryption

The system can be compared to a mailbox into which anyone can put a message, but from which only the person with the key can remove messages.

- Advantages: A message encrypted with a public key can only be decrypted by the owner of the private key. As another (private) key is required for decryption, it is also much harder to work out the decryption key on the basis of large numbers of encrypted messages. This means that the public key does not have to be kept strictly confidential, unlike with symmetric keys.

  Another advantage is easier distribution of public keys. No specially secured channel is required in asymmetric cryptography to transfer the public key from the recipient to the sender encrypting the messages. Less work is thus required in managing the keys than would be the case in symmetric encryption procedures.

- Disadvantages: Complex algorithm (e.g. RSA, named after the three mathematicians Rivest, Shamir and Adleman), and therefore poorer performance than with symmetric encryption.
Encryption processes in practice

In practice, for example with a CPU Web server and Secure Open User Communication, the TLS protocol is used below the relevant application layer. Application layers are HTTP or SMTP, for example, as detailed above.

TLS (Transport Layer Security) uses a combination of asymmetric encryption and symmetric encryption (hybrid encryption) for secure data transfer, for example, over the Internet, and uses the following subprotocols:

- TLS Handshake Protocol, responsible for authentication of communication partners and negotiation of the algorithms and keys to be used for subsequent data transfer on the basis of asymmetric encryption.
- TLS Record Protocol, responsible for encryption of user data with symmetric encryption and data exchange.

Both asymmetric and symmetric encryption are considered secure encryption schemes - there is basically no difference in security between the two procedures. The degree of security depends on parameters such as the selected key length.

Abuse of encryption

You cannot tell what identity is assigned to a public key from the bit string. A fraud could provide their public key and claim to be someone else. If a third party then uses this key thinking that they are addressing their required communication partner, confidential information could end up with the fraud. The fraud then uses their private key to decrypt the message that was not intended for them, and sensitive information falls into the wrong hands.

To prevent this type of abuse, the communication partners must be confident that they are dealing with the right communication partner. This trust is established by using digital certificates in a PKI.
3.6 Secure Communication

3.6.3 Authenticity and integrity through signatures

Attacks from programs that intercept communication between the server and client and act as if they themselves were client or server, are called man-in-the-middle attacks. If the false identity of these programs is not detected, they can obtain important information about the S7 program, for example, or set values in the CPU and attack a machine or plants. Digital certificates are used to avoid such attacks.

Secure communication uses digital certificates that meet the X.509 standard of the International Telecommunication Union (ITU). This allows the identity of a program, a computer or an organization to be checked (authenticated).

How certificates establish trust

The main role of X.509 certificates is to bind an identity with the data of a certificate subject (for example, e-mail address or computer name) to the public key of the identity. Identities can be people, computers or machines.

Certificates are issued by certificate authorities (Certificate Authority, CA) or by the subject of a certificate itself. PKI systems specify how users can trust the certificate authorities and the certificates that they issue.

The certificate process:

1. Anyone wishing to own a certificate submits a certificate application to a registration authority linked to the certificate authority.
2. The certificate authority assesses the application and applicant on the basis of set criteria.
3. If the identity of the applicant can be clearly established, the certificate authority confirms that identity by issuing a signed certificate. The applicant has now become the certificate subject.

The figure below is a simplified overview of the process. It does not show how Alice can check the digital signature.

![Figure 3-9 Signing of a certificate by a certificate authority](image-url)
Self-signed certificates

Self-signed certificates are certificates whose signature comes from the certificate subject and not from an independent certificate authority.

Examples:
- You can create and sign a certificate yourself, for example, to encrypt messages to a communication partner. In the example above, Bob (instead of Twent) could himself sign his certificate with his private key. Using Bob’s public key, Alice can check that the signature and public key from Bob match. This procedure is sufficient for simple internal plant communication that is to be encrypted.
- A root certificate is, for example, a self-signed certificate, signed by the certificate authority (CA), that contains the public key of the certificate authority.

Features of self-signed certificates

The "CN" (Common Name of Subject) for the certificate subject and "Issuer" attributes of self-signed certificates are identical: You have signed your certificate yourself. The field "CA" (Certificate Authority) must be set to "False"; the self-signed certificate should not be used to sign other certificates.

Self-signed certificates are not embedded in a PKI hierarchy.

Certificate content

A certificate to the X.509 V3 standard, the standard that is also used by STEP 7 and the S7-1500 CPUs, consists primarily of the following elements:
- Public key
- Details of the certificate subject (i.e. the holder of the key), for example, the Common Name (CN) of Subject.
- Attributes such as serial number and validity period
- Digital signature from the certificate authority (CA) confirming that the information is correct.

There are also extensions, for example:
- Specification of what the public key may be used for (Key Usage), for example, signing or key encryption.
  - When you create a new certificate with STEP 7, for example in the context of Secure Open User Communication, select the correct entry from the list of possible usages, e.g. "TLS".
- Specification of a Subject Alternative Name (SAN), which is used in secure communication with Web servers (HTTP over TLS), for example, to ensure that the certificate in the address bar of the Web browser also belongs to the Web server specified in the URL.
How signatures are generated and verified

Asymmetric key usage ensures that certificates can be verified: The example of the "MyCert" certificate illustrates the "Sign" and "Verify signature" processes.

Generating a signature:

1. The issuer of the "MyCert" certificate generates a hash value from the certificate data using a specific hash function (for example SHA-1, Secure Hash Algorithm).
   The hash value is a bit string of a constant length. The advantage of the constant length of the hash value is that it always takes the same amount of time to sign.
2. Using the hash value generated in this way and the private key, the issuer of the certificate then generates a digital signature. The RSA signature scheme is often used.
3. The digital signature is saved in the certificate. The certificate is now signed.

Verifying a signature:

1. The authenticator of the "MyCert" certificate obtains the certificate of the issuer and thus the public key.
2. A new hash value is formed from the certificate data with the same hash algorithm that was used for signing (for example SHA-1).
3. This hash value is then compared with the hash value that is determined by means of the public key of the certificate issuer and the signature algorithm for checking the signature.
4. If the signature check produces a positive result, both the identity of the certificate subject as well as the integrity, meaning authenticity and genuineness, of the certificate content are proven. Anyone who has the public key, i.e. the certificate from the certificate authority, can check the signature and thus recognize that the certificate was actually signed by the certificate authority.

The figure below shows how Alice uses the public key in the certificate from Twent (who represents the certificate authority, CA) to verify the signature on Bob's public key. All that is required for verification is therefore the availability of the certificate from the certificate authority at the moment of checking. The validation itself is executed automatically in the TLS session.

![Figure 3-10 Verification of a certificate with the public key of the certificate of a certificate authority](image-url)
**Signing messages**

The method described above for signing and verifying also uses the TLS session for signing and verifying messages.

If a hash value is generated by a message and this hash value is signed with the private key of the sender and attached to the original message, the recipient of the message is able to check the integrity of the message. The recipient decrypts the hash value with the public key of the sender, puts together the hash value from the message received and compares the two values. If the values are not the same, the message has been tampered with on the way.

**Chain of certificates to root certificate**

The certificates of a PKI are often organized hierarchically: The top of the hierarchy is formed by root certificates. Root certificates are certificates that are not signed by a higher-level certificate authority. The certificate subject and certificate issuer of root certificates are identical. Root certificates enjoy absolute trust. They form the "anchor" of trust and must therefore be known to the receiver as trusted certificates. They are stored in an area provided for trusted certificates.

Depending on the PKI, the function of root certificates is, for example, to sign certificates from lower-level certificate authorities, so-called intermediate certificates. This transfers the trust from the root certificate to the intermediate certificate. An intermediate certificate can sign a certificate just like a root certificate; both are therefore referred to as "CA certificates".

This hierarchy can be continued over multiple intermediate certificates until the end-entity certificate. The end-entity certificate is the certificate of the user who is to be identified.

The validation process runs through the hierarchy in the opposite direction: As described above, the certificate issuer is established and the signature checked with the issuer's public key, then the certificate of the higher-level certificate issuer is established along the entire chain of trust to the root certificate.

Conclusion: The chain of intermediate certificates to the root certificate, the certificate path, must be available in every device that is to validate an end-entity certificate of the communication partner, irrespective of the type of secure communication that you configure.

**3.6.4 Managing certificates with STEP 7**

STEP 7 as of version V14 together with the S7-1500-CPU as of firmware version 2.0 support the Internet PKI (RFC 5280) in as far as an S7-1500-CPU is able to communicate with devices that also support the Internet PKI.

The usage of X.509 certificates for verifying certificates as described in the preceding sections, for example, is a result of this.

STEP 7 as of V14 uses a PKI similar to Internet PKI. Certificate Revocation Lists (CRLs), for example, are not supported.
Creating or assigning certificates

You create certificates for various applications in STEP 7 for devices with security properties, such as an S7-1500 CPU as of firmware V2.0.

The following areas in the Inspector window of the CPU allow the creation of new certificates or the selection of existing ones:

- "Protection & Security > Certificate manager" - for the generation and assignment of all types of certificates. TLS certificates for Secure Open User Communication are preset for the generation of certificates.
- "Web server > Security" - for the generation and assignment of Web server certificates.
- "OPC UA > Server > Security" - for the generation and assignment OPC UA certificates.

Figure 3-11 Security settings for an S7-1500 CPU in STEP 7
Special features of the section "Protection & Security > Certificate manager"

Only in this section of the Inspector window do you switch between the global, i.e. project-wide, and the local, i.e. device-specific, certificate manager (option "Use global security settings for the certificate manager"). The option decides whether you have access to all the certificates in the project or not.

- If you do not use the certificate manager in the global security settings, you only have access to the local certificate memory of the CPU. You do not have access, for example, to imported certificates or root certificates. Without these certificates only a restricted functionality is available. You can, for example, only generate self-signed certificates.

- If you use the certificate manager in the global security settings and you are logged on as an administrator, you have access to the global, project-wide certificate memory. You can, for example, assign imported certificates to the CPU, or create certificates that are issued and signed by the project CA (certificate authority of the project).

The figure below shows how the "Global security settings" are shown in the project tree after the "Use global security settings for certificate manager" option has been activated in the Inspector window of the CPU.
When you double-click "User login" in the project tree below the global security settings and log in, a line called "Certificate manager" is displayed, among other data.

When you double-click the "Certificate manager" line, you obtain access to all the certificates in the project, divided into the tabs "CA" (certificate authorities), "Device certificates" and "Trusted certificates and root certificate authorities".
Private keys

STEP 7 generates private keys while generating device certificates and server certificates (end-entity certificates). The location where the private key is stored encrypted depends on the use of the global security settings for the certificate manager:

- If you use global security settings, the private key is stored encrypted in the global (project-wide) certificate memory.
- If you do not use global security settings, the private key is stored encrypted in the local (CPU-specific) certificate memory.

The existence of the private key, which is required to decrypt data, for example, is displayed in the “Private key” column of the “Device certificates” tab of the certificate manager in the global security settings.

When the hardware configuration is loaded, the device certificate, the public key as well as the private key are loaded into the CPU.

<table>
<thead>
<tr>
<th>NOTICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The &quot;Use global security settings for certificate manager&quot; option influences the previously used private key: If you have already created certificates without using the certificate manager in the global security settings and then change the option for using the certificate manager, the private keys are lost and the certificate ID can change. A warning draws your attention to this fact. Therefore specify at the beginning of the project configuration which option is required for the certificate manager.</td>
</tr>
</tbody>
</table>

3.6.5 Examples for the management of certificates.

As explained in the preceding sections, certificates are required for every type of secure communication. The following section shows as an example how you handle the certificates with STEP 7 so that the requirements for Secure Open User Communication are fulfilled.

The devices which are involved at the respective communication partners are differentiated below. The respective steps for supplying the required certificates to the communications participants are described. An S7-1500 CPU or an S7-1500 software controller as of firmware version 2.0 is always required.

The general rule is:

While a secure connection is being established (handshake*), the communication partners as a rule only communicate their end-entity certificates (device certificates).

Therefore the CA certificates required to verify the transmitted device certificate must be located in the certificate memory of the respective communication partner.
Secure Open User Communication between two S7-1500 CPUs

Two S7-1500-CPU, PLC_1 and PLC_2, are to exchange data with each other via Secure Open User Communication.

You generate the required device certificates with STEP 7 and assign them to the CPUs as described below.

STEP 7 project certificate authorities (CA of the project) are used to sign the device certificates.

The certificates are to be referenced by their certificate ID in the user program (TCON communication instruction in combination with the associated system data type, for example TCON_IPV4_SEC). STEP 7 assigns the certificate ID automatically during the generation or creation of certificates.

Procedure

STEP 7 automatically loads the required CA certificates together with the hardware configuration to the participating CPUs so that the requirements for certificate verification exist for both CPUs. You therefore only have to generate the device certificates for the respective CPU; STEP 7 does the rest for you.

1. Mark PLC_1 and activate the "Use global security settings for certificate manager" option in the "Protection & Security" section.

2. Log in as a user in the project tree in the "Global security settings" section. For a new project, the "Administrator" role is planned for the first login.

3. Return to the PLC-1 in the "Protection & Security" section. Click in an empty line in the "Certificate subject" column in the "Device certificates" table to add a new certificate.

4. In the drop-down list for selecting a certificate click the "Add" button. The "Create Certificate" dialog opens.

5. Leave the default settings in this dialog. They are tailored to the usage of Secure Open User Communication (usage: TLS).

   Tip: Supplement the default name of the certificate subject, in this case the CPU name. In order to differentiate you better leave the default CPU name in case you have to manage a large number of device certificates.

   Example: PLC_1/TLS becomes PLC_1-SecOUC-Chassis17FactoryState.

6. Compile the configuration.

   The device certificate and the CA certificate are part of the configuration.

7. Repeat the steps described above for PLC_2.

In the next step you have to create the user programs for the data exchange and load the configurations together with the program.
Using self-signed certificates instead of CA certificates

When creating device certificates you can select the "Self-signed" option. You can create self-signed certificates without being logged in for the global security settings. This procedure is not recommended because the resulting certificates do not exist in the global certificate memory and can therefore not be assigned directly to a partner CPU.

As described above, you should select the name of the certificate subject with care so that the right certificate can be assigned to a device without any doubt.

Verification with the CA certificates of the STEP 7 project is not possible for self-signed certificates. To ensure that self-signed certificates can be verified you have to include the self-signed certificates of the communication partner into the list of trusted partner devices for each CPU. To this purpose you must have activated the "Use global security settings for certificate manager" option and be logged in as a user in the global security settings.

Proceed as follows to add the self-signed certificate of the communication partner of the CPU:

1. Mark PLC_1 and navigate to the "Certificates of partner devices" table in the "Protection & Security" section.
2. Click in an empty line in the "Certificate subject" column in the "Device certificates" table to add a new certificate.
3. Select the self-signed certificate of the communication partner from the drop-down list and confirm the selection.

In the next step you have to create the user programs for the data exchange and load the configurations together with the program.

Secure Open User Communication between S7-1500 CPU as a TLS client and an external device as a TLS server

Two devices are to exchange data with each other via TLS connection or TLS session, for example, exchanging recipes, production data or quality data:

- An S7-1500 CPU (PLC_1) as TLS client; the CPU uses Secure Open User Communication
- An external device, for example a Manufacturing Execution System (MES), as TLS server

The S7-1500 CPU establishes the TLS connection / session to the MES system as TLS client.
The S7-1500 CPU requires the CA certificates of the MES system to authenticate the TLS server: The root certificate and, if appropriate, the intermediate certificates for verifying the certificate path.

You have to import these certificates into the global certificate memory of the S7-1500 CPU. Proceed as follows to import certificates of the communication partner:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the appropriate table (trusted certificates and root certificate authorities) for the certificate to be imported.
3. Right-click in the table to open the shortcut menu. Click "Import" and import the required certificate or the required CA certificates.
   Through the import the certificate has a certificate ID assigned to it and can be assigned to a module in the next step.
4. Mark PLC_1 and navigate to the "Certificates of partner devices" table in the "Protection & Security" section.
5. Click in an empty line in the "Certificate subject" column to add the imported certificates.
6. Select the required CA certificates of the communication partner from the drop-down list and confirm the selection.

Optionally the MES system can also request a device certificate of the CPU to authenticate the CPU (i.e., the TLS client). In this case, the CA certificates of the CPU must be made available to the MES system. The prerequisite for importing the certificates into the MES system is a preceding export of the CA certificates from the STEP 7 project of the CPU. Follow these steps:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the matching table (CA certificate) for the certificate to be exported.
3. Right-click the selected certificate to open the shortcut menu.
4. Click "Export".
5. Select the export format of the certificate.

In the next step you have to create the user programs for the data exchange and load the configurations together with the program.
Secure Open User Communication between an S7-1500 CPU as TLS server and an external device as TLS client

If the S7-1500 CPU acts as TLS server and the external device, for example an ERP system (Enterprise Resource Planning System) establishes the TLS connection / session, you require the following certificates:

- For the S7-1500 CPU, you generate a device certificate (server certificate) with a private key and download it with the hardware configuration into the S7-1500 CPU. You use the "Signed by certificate authority" option when generating the server certificate.

  The private key is required for the key exchange as explained in the figure for the example "HTTP over TLS".

- You have to export the CA certificate of the STEP 7 project for the ERP system and import / load it into the ERP system. With the CA certificate the ERP system verifies the server certificate of the S7-1500 that was transferred from the CPU to the ERP system during the establishment of the TLS connection / session.

![Figure 3-12 Secure OUC between an S7-1500 CPU and ERP system](image)

The required steps are described in the preceding sections.
Secure Open User Communication to a mail server (SMTP over TLS)

An S7-1500 CPU can establish a secure connection to an e-mail server with the communication instruction TMAIL-C.

The system data types TMail_V4_SEC and TMail_QDN_SEC allow you to determine the partner port of the e-mail server and thus to reach the e-mail server via "SMTP over TLS".

Requirement for secure e-mail connection is the importing of the root certificate and the intermediate certificates of the mail server (provider) into the global certificate memory of the S7-1500 CPU. By means of these certificates the CPU can check the server certificate that is sent by the mail server during the establishment of the TLS connection / session.

Proceed as follows to import certificates of the mail server:

1. Open the certificate manager in the global security settings in the project tree.
2. Select the appropriate table (trusted certificates and root certificate authorities) for the certificate to be imported.
3. Right-click in the table to open the shortcut menu. Click "Import" and import the required certificate or the required CA certificates.
   As a result of the import, the certificate has a certificate ID assigned to it and can be assigned to a module in the next step.
4. Mark PLC_1 and navigate to the "Certificates of partner devices" table in the "Protection & Security" section.
5. Click in an empty line in the "Certificate subject" column to add the imported certificates.
6. Select the required CA certificates of the communication partner from the drop-down list and confirm the selection.

In the next step you have to create the user programs for the e-mail client function of the CPU and load the configurations together with the program.
Communications services

3.6 Secure Communication

3.6.6 Example: HTTP over TLS

The following paragraphs show how the mechanisms described are used to establish a secure communication between a Web browser and the Web server of an S7-1500 CPU.

Initially the changes for the "Permit access only with HTTPS" option in STEP 7 are described. As of STEP 7 V14 you have the possibility to influence the server certificate of the Web server of an S7-1500 CPU as of firmware V2.0: The server certificate is generated as of these versions with STEP 7.

In addition it illustrates the processes that are executed when a website of the CPU Web server is called with a Web browser of a PC through an encrypted HTTPS connection.

Using Web server certificates for S7-1500 CPUs, FW V2.0 or higher

For S7-1500 CPUs with a firmware version before V2.0, you were able to set "Permit access only with HTTPS" when setting the Web server properties, without specific requirements applying.

You did not have to handle certificates for these CPUs; the CPU automatically generates the certificates required for the Web server.

For S7-1500 CPUs as of firmware V2.0, STEP 7 generates the server certificate (end-entity certificate) for the CPU. You assign a server certificate to the Web server in the properties of the CPU (Web server > Security).

Because a server certificate name is always preset, there is no change to the easy configuration of the Web server: You activate the Web server. The "Permit access only with HTTPS" option is enabled by default - STEP 7 generates a server certificate with the default name during compiling.

Irrespective of whether you use the certificate manager in the global security settings or not: STEP 7 has all the information required to generate the server certificate.

In addition, you have the possibility to determine the properties of the server certificate, for example, the name or the validity period.
Loading the Web server certificate

The server certificate generated by STEP 7 is then automatically also loaded to the CPU when the hardware configuration is loaded.

- If you use the certificate manager in the global security settings, the certificate authority of the project (CA certificate) signs the server certificate of the Web server: During loading the CA certificate of the project is loaded as well automatically.

- If you do not use the certificate manager in the global security settings, STEP 7 generates the server certificate as a self-signed certificate.

When you address the Web server of the CPU over the IP address of the CPU, a new server certificate (end-entity certificate) must be generated and loaded with each change in the IP address of an Ethernet interface of the CPU. This is necessary because the identity of the CPU changes with the IP address – and the identity requires a signature in accordance with the PKI rules.

You can avoid this problem by addressing the CPU with a domain name instead of its IP address, for example "myconveryer-cpu.room13.myfactory.com". For this purpose, you have to manage the domain names of the CPU via a DNS server.

Supplying a Web browser with a CA certificate of the Web server

In the Web browser the user who accesses the websites of the CPU through HTTPS should install the CA certificate of the CPU. If no certificate is installed, a warning is output recommending that you do not use the page. To view this page, you must explicitly "Add an exception".

The user receives the valid root certificate for download from the "Intro" Web page of the CPU Web server under "Download certificate".

STEP 7 offers a different possibility: Export the CA certificate of the project with the certificate manager into the global security settings in STEP 7. Subsequently import the CA certificate into the browser.
Course of the secure communication

The figure below shows, in simplified terms, how communication is established ("handshake") focusing on the negotiation of keys used for data exchange (here with HTTP over TLS).

However, the course can be applied to all communication options that are based on the usage of TLS, i.e. also for Secure Open User Communication (see Basics for secure communication).

Figure 3-14 Handshake with https

The figure does not show the measures taken at Alice's end (browser) to verify the certificate sent by the Web server. Whether Alice can trust the Web server certificate received and therefore the identity of the Web server, and can accept the exchange of data, depends on positive verification.
The steps for verifying the authenticity of the Web server:

1. Alice must know the public keys of all relevant certificate authorities, which means she requires the complete certificate chain to verify the Web server certificate (i.e. the end-entity certificate of the Web server):

   Alice will generally have the required root certificate in her certificate memory. When a Web browser is installed, a range of trusted root certificates is also installed. If she does not have the root certificate, she has to download it from the certificate authority and install it in the certificate store of the browser. The certificate authority can also be the device on which the Web server is located.

   You have the following options for obtaining the intermediate certificates:
   - The server itself sends the required intermediate certificates to Alice along with its end-entity certificate – in the form of a signed message so that Alice can verify the integrity of the certificate chain.
   - The certificates often contain the URLs of the certificate issuer. Alice can load the required intermediate certificates from these URLs.

   When you work with certificates in STEP 7 it is always assumed that you have imported the intermediate certificates and the root certificate into the project and assigned them to the module.

2. Alice validates the signatures in the certificate chain with the public keys of the certificates.

3. The symmetric key must be generated and transferred to the Web server.

4. If the Web server is addressed by its domain name, Alice also verifies the identity of the Web server in accordance with the Internet PKI rules defined in RFC 2818. She is able to do this because the URL of the Web server, in this case the "Fully Qualified Domain Name" (FQDN), is saved in the end-entity certificate of the Web server. If the certificate entry in the "Subject Alternative Name" field corresponds to the entry in the address bar of the browser, everything is fine.

   The process continues with the exchange of data with the symmetric key, as shown in the figure above.
3.7 SNMP

3.7.1 Disabling SNMP

The network management protocol SNMP (Simple Network Management Protocol) is a protocol that uses various services and tools for detection and diagnostics of the network topology.

Which SNMP requests the S7-1500 CPUs and the S7-1200 CPUs can receive, is described in this FAQ [https://support.industry.siemens.com/cs/ww/en/view/79993228].

SNMP uses the transport protocol UDP. SNMP recognizes two network components, the SNMP manager and the SNMP client. The SNMP manager monitors the network nodes: The SNMP clients collect the various network-specific information in the individual network nodes and store it in a structured form in the MIB (Management Information Base). Various services and tools can run detailed network diagnostics with the help of these data.

Under certain conditions, it is useful to disable SNMP. Examples:

- The security guidelines in your network do not allow the use of SNMP.
- You use your own SNMP solution, e.g. with your own communications instructions.

If you disable SNMP for a device, various options for diagnostics of the network topology (e.g. using the PRONETA tool or the Web server of the CPU) are no longer available to you.

Disabling SNMP

To disable SNMP for one of the integrated interfaces of a S7-1500 CPU, follow these steps:

1. In STEP 7, create a data block that contains the structure of data record B071H.
   - The following table shows the structure of the data record B071H.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Element</th>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>BlockID</td>
<td>F003H</td>
<td>Header</td>
</tr>
<tr>
<td>2-3</td>
<td>BlockLength</td>
<td>8</td>
<td>The data record length is counted starting at byte 4 &quot;Version&quot;.</td>
</tr>
<tr>
<td>4</td>
<td>Version</td>
<td>01H</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Subversion</td>
<td>00H</td>
<td></td>
</tr>
<tr>
<td>6-7</td>
<td>Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>8-11</td>
<td>SNMP controller</td>
<td>Disable/enable SNMP</td>
<td>If you want to disable SNMP, enter the value 0. If you want to enable SNMP, enter the value 1.</td>
</tr>
</tbody>
</table>

2. Transfer the data record B071H in the startup OB (OB100) with the WRREC instruction (write data record) to the CPU.
   Use the hardware ID of an integrated interface of the CPU here.
3.7.2 Example: Disabling SNMP for a CPU 1516-3 PN/DP

Task

As the security guidelines in your network do not allow SNMP, you want to disable SNMP for a CPU 1516-3 PN/DP.

Requirements

- CPU 1516-3 PN/DP with firmware version V2.0
- STEP 7 as of V14

Solution

First, create a data block that contains the structure of data record B071H. The figure below shows the data block "Deactivate SNMP". The data block "Deactivate SNMP" contains the data record B071H as well as additional tags that you use to transfer the data record. The tag "snmp_deactivate" is used to trigger the job for WRREC. Place this tag in the retentive memory area so that the value is also available in the startup OB (OB100).

Transfer the data record B071H in the startup OB (OB100) to CPU 1516-3 PN/DP with the WRREC instruction (write data record).
In the following program code, the data record B071H is transferred with the WRREC instruction in a REPEAT UNTIL loop.

ORGANIZATION_BLOCK "Startup"
TITLE = "Complete Restart"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
BEGIN
  REPEAT
    "WRREC_DB_1"
    (REQ := "Deactivate SNMP".snmp_deactivate,
     //Transfer data record
     INDEX:=16#B071,
     //Data record number for SNMP deactivation
     ID:="Local~PROFINET_interface_1",
     //any integrated PROFINET Interface
     DONE => "Deactivate SNMP".snmp_done,
     ERROR => "Deactivate SNMP".snmp_error,
     STATUS => "Deactivate SNMP".snmp_status,
     RECORD := "Deactivate SNMP".snmp_record)
     //Data record
    UNTIL "Deactivate SNMP".snmp_done OR "Deactivate SNMP".snmp_error
  END_REPEAT;
END_ORGANIZATION_BLOCK

Using program code

You will find the full program code here.

Follow these steps to apply the program code to your project:

1. Copy the entire program code to the clipboard with Ctrl+A, Ctrl+C.
2. Open a text editor (e.g. "Editor").
3. Paste the content of the clipboard to the text editor with Ctrl+V.
4. Save the document as an scl file, e.g. SNMP_DEACT.scl.
5. Open your project in STEP 7.
6. Import the scl file as an external source.
   You will find further information on importing external sources in the STEP 7 online help.
7. Generate the startup OB and the data blocks. (right-click on the scl file, shortcut menu:
   "Generate blocks from source")

Re-enabling SNMP

With small changes, you can use the program code used above to enable SNMP.

In the user program, assign the "Deactivate SNMP".snmp_record.SNMPControl tag the value "1":
"Deactivate SNMP".snmp_record.SNMPControl := 1;

SNMP will then be enabled again the next time the CPU is started.
PG communication

Properties

Using PG communication, the CPU or another module capable of communication exchanges data with an engineering station (for example PG, PC). The data exchange is possible via PROFIBUS and PROFINET subnets. The gateway between S7 subnets is also supported.

PG communication provides functions needed to load programs and configuration data, run tests, and evaluate diagnostic information. These functions are integrated in the operating system of the module capable of communication.

A PG/PC can be connected to a CPU online. The PG/PC can operate a maximum of 4 online connections at one time (for example to 4 CPUs).

Requirements

- The PG/PC is physically connected to the communication-capable module.
- If the communication-capable module is to be reached via S7 routing, the hardware configuration has to be loaded in the participating stations (S7 router and end point).

Procedure for connecting online

You must establish an online connection to the CPU for the programming device communication:

1. Select the CPU in the project tree in STEP 7.
2. Select the "Online > Go online" menu command.
3. In the "Go online" dialog, make the following settings for your online connection:
   - Select interface type (e.g. PN/IE) in the "Type of PG/PC interface" drop-down list.
   - In the "PG/PC interface" drop-down list, select the PG/PC interface (e.g. Ind. Ethernet card) you want to use to establish the online connection.
   - Select the interface or the S7 subnet with which the programming device/PC is physically connected from the "Connection to interface/subnet" drop-down list.
If the communication-capable module can be reached via an S7 router (gateway), select the S7 router that connects the subnets in question from the "1st gateway" drop-down list.

4. Click "Start search".
   All devices that you can address with PG communication appear shortly thereafter in the table "Compatible devices in target subnet".
5. In the "Compatible devices in target subnet" table, select the relevant CPU and confirm with "Go online".

Additional information

You can find more information on "Go online" in the STEP 7 online help.
HMI communication

Properties

Using HMI communication, one or more HMI devices (for example HMI Basic/Comfort/Mobile Panel) exchanges data with a CPU for operator control and monitoring with via the PROFINET or PROFIBUS DP interface. The data exchange is via HMI connections.

If you want to set up several HMI connections to a CPU, use for example:

- The PROFINET and PROFIBUS DP interfaces of the CPU
- CPs and CMs with the relevant interfaces

Procedure for setting up HMI communication

As soon as you drag-and-drop a tag, for example a tag from a global data block into an HMI screen or into the HMI tag table, STEP 7 automatically sets up an HMI connection. Alternatively, you can also set up the HMI connection yourself.

To set up an HMI connection, follow these steps.

1. Configure the HMI device in an existing configuration with a CPU in the network view of the Devices & networks editor of STEP 7.
2. Select the "Connections" button and then "HMI connection" from the drop-down list.
3. Drag-and-drop a line between the end points of the connection (HMI device and CPU). The end points are highlighted in color. If the required S7 subnet does not yet exist, it is created automatically.
4. In the "Connections" tab, select the row of the HMI connection.
   In the "General" area of the "Properties" tab, you see the properties of the HMI connection, some of which you can change.

![HMI connection setup](image)

Figure 5-1 Setting up HMI communication

5. Download the hardware configuration to the CPU.
6. Download the hardware configuration to the HMI device.

Additional information

You can find information on S7 routing for HMI connections in the section **S7 Routing** (Page 260).

You can find more information on setting up HMI connections in the STEP 7 online help.
6.1 Overview of Open User Communication

Features of Open User Communication

Through Open User Communication, also called "open communication", the CPU exchanges data with another device capable of communication. Open User Communication has the following features and characteristics:

- Open standard (communication partners can be two SIMATIC CPUs or a SIMATIC CPU and a suitable third-party device).
- Communication via various protocols (in STEP 7 known as "Connection types")
- High degree of flexibility in terms of the data structures transferred; this allows open data exchange with any communications devices as long as these support the connection types available.
- Secure Communication: To protect your automation system, you can exchange data securely over Open User Communication. With Secure Open User Communication, the data is sent signed and encrypted.
- Open User Communication is possible in various automation systems, see technical specifications of the respective manuals.

Examples:
- Integrated PROFINET / Ind. Ethernet interfaces of CPUs (S7-1500, ET 200SP CPU, S7-1500 Software Controller, CPU 1516pro-2 PN)
- PROFINET / Ind. Ethernet interfaces of communications modules (for example CP 1543-1, CM 1542-1)

Information on Secure Open User Communication is available in the section Secure Communication (Page 37).

Information on S7-1500R/H

You can find information on Open User Communication with the S7-1500R/H redundant system in section Communication with the redundant system S7-1500R/H (Page 283).
6.2 Protocols for Open User Communication

The following protocols are available for open communication:

### Table 6-1 Transport protocols for open communication

<table>
<thead>
<tr>
<th>Transport protocol</th>
<th>Via interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP according to RFC 793</td>
<td>PROFINET/Industrial Ethernet</td>
</tr>
<tr>
<td>ISO-on-TCP according to RFC 1006 (Class 4)</td>
<td>PROFINET/Industrial Ethernet</td>
</tr>
<tr>
<td>ISO according to ISO/IEC 8073</td>
<td>Industrial Ethernet (only CP 1543-1)</td>
</tr>
<tr>
<td>UDP according to RFC 768</td>
<td>PROFINET/Industrial Ethernet</td>
</tr>
<tr>
<td>FDL</td>
<td>PROFIBUS</td>
</tr>
</tbody>
</table>

### Table 6-2 Application protocols for open communication

<table>
<thead>
<tr>
<th>Application protocol</th>
<th>Used transport protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus TCP</td>
<td>TCP according to RFC 793</td>
</tr>
<tr>
<td>E-mail</td>
<td>TCP according to RFC 793</td>
</tr>
<tr>
<td>FTP</td>
<td>TCP according to RFC 793</td>
</tr>
</tbody>
</table>

TCP, ISO-on-TCP, ISO, UDP

Prior to data transfer, these protocols (except UDP) establish a transport connection to the communications partner. Connection-oriented protocols are used when potential loss of data needs to be avoided.

The following is possible with UDP:

- Unicast to one device or broadcast to all devices on PROFINET via the PROFINET interface of the CPU or the Industrial Ethernet interface of the CP 1543-1
- Multicast to all recipients of a multicast group via the PROFINET interface of the CPU* or the PROFINET / Industrial Ethernet interface of the CP 1543-1

* As of firmware version V2.0, the PROFINET interface of the CPU supports a maximum of 5 multicast groups

Maximum user data lengths UDP: Which maximum user data length is supported for the UDP is described in the technical specifications in the respective manuals.
Protocol for communication via PROFIBUS: FDL

Data transfer via an FDL connection (Fieldbus Data Link) is suitable for the transfer of related blocks of data to a communications partner on PROFIBUS that supports the sending and receiving of data according to the FDL service SDA (Send Data with Acknowledge) according to EN 50170, Vol 2. Both partners have the same rights; in other words, each partner can initiate sending and receiving event-driven.

In keeping with the FDL service SDN (Send Data with No Acknowledge) according to EN 50170, Vol 2, the following is possible with FDL:

- Broadcast to all devices on PROFIBUS via the PROFIBUS interface of the CM 1542-5
- Multicast to all recipients of a multicast group via the PROFIBUS interface of the CM 1542-5

Modbus TCP

The Modbus protocol is a communication protocol with linear topology based on a master/slave architecture. In the Modbus TCP (Transmission Control Protocol), the data is transmitted as TCP/IP packets.

Communication is controlled solely by suitable instructions in the user program.

E-mail and FTP

You can use email to send for example, data block contents (e.g. process data) as an attachment.

You can use the FTP connection (FTP = File Transfer Protocol) to transmit files to and from S7 devices.

The communication is controlled by instructions in the user program at the client end.

Application example: MQTT Publisher for the SIMATIC S7-1500 CPU

The "Message Queue Telemetry Transport" (MQTT) is a simple protocol on the TCP/IP level. It is suitable for the exchange of messages between devices with lower functionality and for the transfer via unreliable networks.

The application example provides a function block with which you can implement the MQTT protocol into the SIMATIC S7-1500.

You can find the application example on the Internet [https://support.industry.siemens.com/cs/ww/en/view/109748872].
Block library for SYSLOG messages

Syslog is a simply structured binary profile on UDP/IP level. It enables applications to send messages, warnings or error states to a Syslog server. Syslog is typically used for computer system management and security monitoring, and has established itself as a standard in the field of protocols.

The "LSyslog" library offers you a solution to implement the Syslog protocol in an S7-1500. In addition to the library, an application example is provided that shows you how to generate Syslog messages in your controller and send them to the Syslog server.

You can find the block library "LSyslog" and the associated application example on the Internet [https://support.industry.siemens.com/cs/ww/en/view/51929235].

6.3 Instructions for Open User Communication

Introduction

You set up Open User Communication via the corresponding connection (for example, TCP connection) as follows:

- By programming in the user programs of the communications partners or
- By configuring the connection in STEP 7 in the hardware and network editor

Regardless of whether you set up the connection by programming or configuring, instructions are always required in the user programs of both communications partners for sending and receiving the data.
Setting up the connection via the user program

If the connection is set up by programming, the connection establishment and termination is implemented using instructions in the user program.

In certain areas of application it is an advantage not to set up the communications connections statically by configuring in the hardware configuration, but to have them set up by the user program. You can set up the connections via a specific application program-controlled and therefore when necessary. Programmed connection setup also allows connection resources to be released following data transfer.

A data structure is necessary for each communications connection that contains the parameters for establishing the connection (for example system data type "TCON_IP_v4" for TCP).

The system data types (SDT) are provided by the system and have a predefined structure that cannot be changed.

The various protocols have their own data structures (see table below). The parameters are stored in a data block ("connection description DB") for example of the system data type TCON_IP_v4.

There are two ways in which you can specify the DB with the data structure:

- Recommendation: Have the data block created automatically in the properties in the program editor during parameter assignment of the connection for the TSEND_C, TRCV_C and TCON instructions.
- Create the data block manually, assign parameters to it and write it directly to the instruction

Necessary for:
- Secure OUC
- Connection over DNS
- E-mail
- FTP

You can modify the connection parameters in the "connection description DB".

This FAQ [https://support.industry.siemens.com/cs/ww/en/view/58875807] describes how to program the TCON instruction to set up a connection for Open User Communication between two S7-1500 CPUs.
### Protocols, system data types and employable instructions for programmed setup

The following table shows the protocols of the Open User Communication and the matching system data types and instructions.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>System data type</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>• TCON_QDN</td>
<td>Establish connection and send/receive data via:</td>
</tr>
<tr>
<td></td>
<td>• TCON_IP_v4</td>
<td>• TSEND_C/TRCV_C or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TCON, TSEND/TRCV or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TCON, TUSEND/TURCV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td>ISO-on-TCP</td>
<td>• TCON_IP_RFC</td>
<td></td>
</tr>
<tr>
<td>ISO according to ISO/IEC 8073 (Class 4)</td>
<td>• TCON_ISOnative(^1)</td>
<td>Establish connection and send/receive data via:</td>
</tr>
<tr>
<td></td>
<td>• TCON_Configured</td>
<td>• TSEND_C/TRCV_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TUSEND/TURCV/TRCV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td>UDP</td>
<td>• TCON_IP_v4</td>
<td>Establish connection and send/receive data via:</td>
</tr>
<tr>
<td></td>
<td>• TADDR_Param</td>
<td>• TSEND_C/TRCV_C</td>
</tr>
<tr>
<td></td>
<td>• TADDR_SEND_QDN</td>
<td>• TUSEND/TURCV/TRCV</td>
</tr>
<tr>
<td></td>
<td>• TADDR_RCV_IP</td>
<td>(connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td>FDL(^1)</td>
<td>• TCON_FDL</td>
<td>Establish connection and send/receive data via:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TSEND_C/TRCV_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TCON, TSEND/TRCV or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TCON, TUSEND/TURCV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td>Modbus TCP</td>
<td>• TCON_IP_v4</td>
<td>• MB_CLIENT</td>
</tr>
<tr>
<td></td>
<td>• TCON_QDN</td>
<td>• MB_SERVER</td>
</tr>
<tr>
<td>E-mail</td>
<td>• TMAIL_v4</td>
<td>• TMAIL_C</td>
</tr>
<tr>
<td></td>
<td>• TMAIL_v6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TMAIL_FQDN</td>
<td></td>
</tr>
<tr>
<td>FTP(^2)</td>
<td>• FTP_CONNECT_IPV4(^3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• FTP_CONNECT_IPV6(^3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• FTP_CONNECT_NA_ME(^3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FTP_CMD</td>
</tr>
</tbody>
</table>

\(^1\) This protocol can only be used with the CM 1542-5
\(^2\) This protocol can only be used with the CP 1543-1
\(^3\) User-defined data type
The following table shows you the different connections of the Secure Open User Communication and the matching system data types and instructions.

<table>
<thead>
<tr>
<th>Secure OUC connection</th>
<th>System data type</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure TCP connection from an S7-1500 CPU as TLS client to a third-party PLC (TLS server)</td>
<td>TCON_QDN_SEC</td>
<td>TSEND_C/TRCV_C, TCON</td>
</tr>
<tr>
<td>Secure TCP connection from an S7-1500 CPU as TLS server to a third-party PLC (TLS client)</td>
<td>TCON_IP_V4_SEC</td>
<td></td>
</tr>
<tr>
<td>Secure TCP connection between two S7-1500 stations</td>
<td>TCON_QDN_SEC1</td>
<td></td>
</tr>
<tr>
<td>Secure connection to a mail server²</td>
<td>TMAIL_V4_SEC, TMAIL_QDN_SEC</td>
<td>TMAIL_C (V5.0 or higher)</td>
</tr>
<tr>
<td>Secure Modbus TCP connection</td>
<td>TCON_IP_V4_SEC1, TCON_QDN_SEC</td>
<td>MB_Client, MB_Server</td>
</tr>
</tbody>
</table>

¹ Also possible for CP 1543-1
² Secure connection to a mail server also possible with CP1543-1 und TMAIL_C (V4.0)

**Setting up the connection with connection configuration**

When setting up through the configuration of the connection, the address parameters of the connection are specified in the hardware and network editor of STEP 7.

To send and receive the data, use the same instructions as when the connections are set up by programming:

**Table 6-4 Instructions for sending/receiving with configured connections**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Send/receive with configured connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported instructions:</td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>Send/receive data via:</td>
</tr>
<tr>
<td>ISO-on-TCP</td>
<td>• TSEND_C/TRCV_C or</td>
</tr>
<tr>
<td>ISO according to ISO/IEC 8073 (Class 4)</td>
<td>• TSEND/TRCV or</td>
</tr>
<tr>
<td></td>
<td>• TUSEND/TURCV</td>
</tr>
<tr>
<td>UDP</td>
<td>Send/receive data via:</td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C or</td>
</tr>
<tr>
<td></td>
<td>• TUSEND/TURCV</td>
</tr>
<tr>
<td>FDL</td>
<td>Send/receive data via:</td>
</tr>
<tr>
<td></td>
<td>• TSEND_C/TRCV_C or</td>
</tr>
<tr>
<td></td>
<td>• TSEND/TRCV or</td>
</tr>
<tr>
<td></td>
<td>• TUSEND/TURCV</td>
</tr>
<tr>
<td>Modbus TCP</td>
<td>Not supported</td>
</tr>
<tr>
<td>E-mail</td>
<td>Not supported</td>
</tr>
<tr>
<td>FTP</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Additional instructions for open communication

You can use the following instructions for connections set up in the user program as well as for configured connections:

- T_RESET: Terminating and establishing a connection
- T_DIAG: Check the connection

Basic examples for Open User Communication

The Siemens Online Support offers you function blocks (FBs) that facilitate the handling of the instructions of the Open User Communication. You can find the function block with corresponding examples on the Internet: [https://support.industry.siemens.com/cs/ww/en/view/109747710](https://support.industry.siemens.com/cs/ww/en/view/109747710).

Additional information

The STEP 7 online help describes:

- The user and system data types
- The instructions for open communication
- The connection parameters

You will find information about the allocation and release of connection resources in the section Allocation of connection resources (Page 271).

See also

Secure Open User Communication (Page 94)
6.4 Open User Communication with addressing via domain names

As of firmware version V2.0, S7-1500 CPUs, ET 200SP CPUs and the CPU 1516pro-2 PN support Open User Communication with addressing via Domain Name System (DNS). A DNS client is integrated in the CPU. In the case of communication via DNS, you use domain names as an alias for IP addresses to address communication partners. Addressing of the communication partners via domain names is possible for open communication via TCP and UDP.

At least one DNS server must be located in your network as a requirement for communication via DNS.

The S7-1500 software controller supports communication via DNS for all interfaces that are assigned to the software controller.

Setting up communication via DNS

The DNS client of the CPU must know the IPv4 address of at least one DNS server so that a CPU can establish a connection to a communication partner via its domain name. The CPU supports up to 4 different DNS servers.

To set up communication via domain names for an S7-1500 CPU, follow these steps:

1. Select the CPU in the network view of STEP 7.
2. In the Inspector window go to "Properties" > "General" > "DNS configuration".
3. Enter the IPv4 address of a DNS server in the "DNS server addresses" column of the "Server list" table.
   You can enter up to 4 IPv4 addresses of DNS servers.

![Figure 6-1 Entering DNS server addresses using a CPU 1516-3 PN/DP as an example](image)
Setting up a TCP connection via the domain name of the communication partner

For TCP communication via the domain name you need to create a data block with the TCON_QDN system data type yourself, assign parameters and call it directly at the instruction. The TCON, TSEND_C and TRCV_C instructions support the system data type TCON_QDN:

To set up a TCP connection via the domain name of the communication partner, follow these steps:

1. Create a global data block in the project tree.
2. Define a tag of the data type TCON_QDN in the global data block.

The example below shows the global data block "Data_block_1" in which the tag "DNS Connection1" of data type TCON_QDN is defined.

3. Program the parameters of the TCP connection (for example the fully qualified domain name (FQDN)) in the tag of data type TCON_QDN.
4. Create a TCON instruction in the program editor.
5. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_QDN.

In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag "DNS connection1" (data type TCON_QDN).
Addressing a UDP connection via the domain name of the communication partner

For S7-1500 CPUs as of firmware version V2.0, you can address the recipient with its fully qualified domain name (FQDN) when sending data via UDP. With the instruction TUSEND at the parameter ADDR, you hereby reference a structure of the type TADDR_SEND_QDN.

The receiver can return an IPv4 or an IPv6 address. With the TURCV instruction at the ADDR parameter, you therefore reference a structure of the TADDR_RCV_IP type. Only this structure can include both IP address types.

Note

Network load

In contrast to the TCP the UDP protocol does not work connection-oriented. For every edge at the block parameter REQ, the TUSEND or TURCV command performs queries of the DNS server. This can lead to high network load or load on the DNS server.

Additional information

You can find more information about the system data types TCON_QDN, TADDR_SEND_QDN and TADDR_RCV_IP in the STEP 7 online help.

How to set up a secure TCP connection via the domain name of the communication partner is described in the section Secure Open User Communication [Page 94].

6.5 Setting up Open User Communication via TCP, ISO-on-TCP, UDP and ISO

Configuring a connection for the TSEND_C, TRCV_C or TCON instructions

Requirement: A TSEND_C, TRCV_C or TCON instruction is created in the programming editor.

1. Select a TCON, TSEND_C or TRCV_C block of Open User Communication in the program editor.

2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.

The connection parameters already known are displayed:
- Name of the local end point
- Interface of the local end point
- IPv4 address of the local end point

Figure 6-4  Connection parameters for TSEND_C

4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communication partner. Certain connection parameters are then entered automatically.

The following parameters are set:
- Name of the partner end point
- Interface of the partner end point
IPv4 address of the partner end point
If the connection partners are networked, the name of the subnet is displayed.

5. In the "Configuration type" drop-down list, select between using program blocks or configured connections.

6. Select an existing connection description DB in the "Connection data" drop-down list or for configured connections select an existing connection under "Connection name". You can also create a new connection description DB or a new configured connection. Later, you can still select other connection description DBs or configured connections or change the names of the connection description DBs in order to create new data blocks:
   - You can also see the selected data block at the interconnection of the CONNECT input parameter of the selected TCON, TSEND_C or TRCV_C instruction.
   - If you have already specified a connection description DB for the connection partner using the CONNECT parameter of the TCON, TSEND_C or TRCV_C instruction, you can either use this DB or create a new DB.
   - If you edit the name of the displayed data block in the drop-down list, a new data block with the changed name but with the same structure and content is generated and used for the connection.
   - Changed names of a data block must be unique in the context of the communication partner.
   - A connection description DB must have the structure TCON_Param, TCON_IP_v4 or TCON_IP_RFC, depending on CPU type and connection.
   - A data block cannot be selected for an unspecified partner.

Additional values are determined and entered after the selection or creation of the connection description DB or configured connection.

The following is valid for specified connection partners:
   - ISO-on-TCP connection type
   - Connection ID with default of 1
   - Active connection establishment by local partner
   - TSAP ID
     for S7-1200/1500: E0.01.49.53.4F.6F.6E.54.43.50.2D.31

The following is valid for unspecified connection partners:
   - TCP connection type
   - Partner port 2000

The following applies for a configured connection with a specified connection partner:
   - TCP connection type
   - Connection ID with default of 257
   - Active connection establishment by local partner
   - Partner port 2000

The following applies for a configured connection with an unspecified connection partner:
   - TCP connection type
   - Local port 2000
7. Enter a connection ID as needed for the connection partner. No connection ID can be assigned to an unspecified partner.

**Note**
You must enter a unique value for the connection ID at a known connection partner. The uniqueness of the connection ID is not checked by the connection parameter settings and there is no default value entered for the connection ID when you create a new connection.

8. Select the desired connection type in the relevant drop-down list. Default values are set for the address details depending on the connection type. You can choose between the following:
   - TCP
   - ISO-on-TCP
   - UDP
   - ISO (only with Configuration mode "Use configured connection")

   You can edit the input boxes in the address details. Depending on the selected protocol, you can edit the ports (for TCP and UDP) or the TSAPs (for ISO-on-TCP and ISO).

9. Use the "Active connection establishment" check box to set the connection establishment characteristics for TCP, ISO and ISO-on-TCP. You can decide which communication partner establishes the connection actively.

   Changed values are checked immediately for input errors by the connection configuration and entered in the data block for the connection description.

**Note**
Open User Communication between two communication partners can only work when the program section for the partner end point has been downloaded to the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.
Configuring connections, e.g. for TSEND/TRCV

If you want to use the instructions for TSEND/TRCV for open communication, for example, you first need to configure a connection (e.g. TCP connection).

To configure a TCP connection, follow these steps:

2. Click the "Connections" button and select the "TCP connection" connection type from the drop-down list.
3. Using drag-and-drop, connect the communication partner with each other (via an interface or local end point). If the required S7 subnet does not yet exist, it is created automatically.

You can also set up a connection to unspecified partners.

4. Select the created connection in the network view.
5. Set the properties of the connection in the "Properties" tab in the "General" area, for example the name of the connection and the interfaces of the communications partner that will be used.

For connections to an unspecified partner, set the address of the partner. You can find the local ID (reference of the connection in the user program) in the "Local ID" area.

6. In the Project tree, select the "Program blocks" folder for one of the CPUs and open OB1 in the folder by double-clicking on it. The program editor opens.
7. Select the required instruction from the "Instructions" task card, "Communication" area, "Open user communication", for example TSEND and drag it to a network of OB1.
8. At the ID parameter of the instruction, assign the local ID of the configured connection to be used for the transmission of data.
9. Interconnect the "DATA" parameter of the TSEND instruction with the user data, for example in a data block.

10. Download the hardware configuration and user program to the CPU.

Based on the procedure described above, set up the connection on the partner CPU with the instruction for receiving, TRCV, and download it to the CPU.
Open User Communication

6.5 Setting up Open User Communication via TCP, ISO-on-TCP, UDP and ISO Communication

Point to note with ISO connections with CP 1543-1

If you use the "ISO connection" connection type, you will need to select the "Use ISO protocol" check box in the properties of the CP so that addressing using MAC addresses will work.

Figure 6-5  Select CP 1543-1 ISO protocol

Additional information

The STEP 7 online help describes:
- The instructions for open communication
- The connection parameters

This FAQ [https://support.industry.siemens.com/cs/ww/en/view/109479564] describes how the instructions TSEND_C and TRCV_C behave in the S7-1500.
6.6 Setting up communication over FDL

Requirements

- Configuration software: STEP 7 Professional V14
- End point of the connection: CPU S7-1500 firmware version V2.0 or higher with communication module CM 1542-5 with firmware version V2.0

Setting up a configured FDL connection

Proceed as follows to set up a configured FDL connection in STEP 7:

1. Create a TSEND_C instruction in the program editor.
2. Select the TSEND_C instruction and go to "Properties" > "General" > "Connection parameters" in the Inspector window.
3. Under End point, select the partner end point. Use one of the two partner end points below:
   - CPU S7-1500 with CM 1542-5
   - Unspecified
4. Under Configuration type, select "Use configured connection".
5. Under Connection type, select "FDL".
6. Under Interface, select the following interfaces:
   - Local: PROFIBUS interface of CM 1542-5
   - Specified partner: PROFIBUS interface of CM 1542-5
7. Under Connection data, select the setting <new>. 
The figure below shows a fully configured FDL connection in STEP 7.

![Figure 6-6 Configuring the FDL connection](image)

**Setting up an FDL connection in the user program**

For communication via FDL, you need to create the data block of the TCON_FDL system data type yourself in each case, assign parameters and call it directly at the instruction. Follow these steps:

1. Create a global data block in the project tree.
2. In the global data block, define a tag of the data type TCON_FDL.
3. Program the parameters of the FDL connection (e.g. the PROFIBUS addresses) in the tag of the data type TCON_FDL.

![Figure 6-7 Programming an FDL connection](image)
4. Create a TCON instruction in the program editor.

5. Interconnect the CONNECT parameter of the TCON instruction with the tag of the data type TCON_FDL.

In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag “FDL_Connection” (data type TCON_FDL).

![Example: TCON Instruction for FDL connection](image-url)
6.7 Setting up communication with Modbus TCP

Setting up a connection for Modbus TCP via the user program

The parameter assignment takes place in the program editor at the instruction MB_CLIENT or MB_SERVER.

Procedure for setting up communication using Modbus TCP

The MB_CLIENT instruction communicates as a Modbus TCP client via the TCP connection. You establish a connection between the client and the server with the instruction, send Modbus requests to the server and receive the corresponding Modbus responses. You also control the setup of the TCP connection with this instruction.

The MB_SERVER instruction communicates as a Modbus TCP server via the TCP connection. The instruction processes connection requests of a Modbus client, receives and processes Modbus requests and sends responses. You also control the setup of the TCP connection.

Requirement: The client can reach the server via IP communication in the network.


2. In the Project tree, select the "Program blocks" folder and open OB1 in the folder by double-clicking on it. The program editor opens.

3. Select the required instruction, for example MB_CLIENT, from the "Instructions" task card, "Communication" area, "Other", "MODBUS TCP" and drag it to a network of OB1.
4. Assign the parameters of the MB_CLIENT or MB_SERVER instruction. Observe the following rules:

An IPv4 server address must be specified for each MB_CLIENT connection.

Each MB_CLIENT or MB_SERVER connection must use a unique instance DB with one of the data structures TCON_IP_v4 or TCON_QDN.

Each connection requires a unique connection ID. The connection ID and instance DB belong together in pairs and must be unique for each connection.

5. Download the hardware configuration and user program to the CPU.
6.7 Setting up communication with Modbus TCP

**Modbus TCP server as gateway to Modbus RTU**

If you use a Modbus TCP server as a gateway to a Modbus RTU protocol, address the slave device in the serial network using the static parameter, MB_UNIT_ID. The MB_UNIT_ID parameter corresponds to the field of the slave address in the Modbus RTU protocol. The MB_UNIT_ID parameter in this case would forward the request to the correct Modbus RTU slave address.

You do not have to program the gateway function yourself.

You can find the MB_UNIT_ID parameter in the instance data block associated with MB_CLIENT instruction.

You can find more information on the MB_UNIT_ID parameter in the STEP 7 online help.

**Reference**

- This FAQ [https://support.industry.siemens.com/cs/ww/en/view/94766380](https://support.industry.siemens.com/cs/ww/en/view/94766380) describes how to program and configure the Modbus TCP communication between two S7-1500 CPUs.

- This FAQ [https://support.industry.siemens.com/cs/ww/en/view/102020340](https://support.industry.siemens.com/cs/ww/en/view/102020340) describes how to program and configure Modbus TCP communication between an S7-1500 CPU and an S7-1200 CPU.
6.8 Setting up communication via e-mail

Setting up a connection for e-mail via the user program

For communication using e-mail, you need to create the data block of the relevant system data type yourself, assign parameters and call the instruction directly. This procedure is introduced below.

Procedure for setting up communication using e-mail

A CPU can send e-mails. To send e-mails from the user program of the CPU, use the TMAIL_C instruction.

Requirement: The SMTP server can be reached via the IPv4 network.

2. Assign parameters to the instruction TMAIL_C, for example enter the subject of the e-mail in Subject.
3. In a global data block create a variable of the type TMAIL_v4, TMAIL_v6 (only CP 1543-1) or TMAIL_FQDN (only CP 1543-1).
4. Set the connecting parameters of the TCP connection in the variable in the "Start value" column. Enter the IPv4 address of the mail server, for example, for the "MailServerAddress" (for TMAIL_v4)

Note

Connection parameters InterfaceId and ID

Note that you can enter the value "0" for the interface ID and the ID with instruction version V5.0 or higher of the instruction TMAIL_C in the data type TMAIL_V4_SEC. In this case the CPU itself searches for a matching local CPU interface or a free connection ID.

Interconnect the variable to the MAIL_ADDR_PARAM parameter of the TMAIL_C instruction.

5. Download the hardware configuration and user program to the CPU.

Additional Information

The STEP 7 online help describes:

- The system data types
- The instructions for open communication
- The connection parameters
6.9 Setting up communication via FTP

Setting up a connection for FTP via the user program

For communication via FTP, you need to create the data block of the relevant system data type yourself, assign parameters and call the instruction directly. This procedure is introduced below.

FTP client and server functionality

Files can be sent by a CPU to an FTP server and can be received from the FTP server. Communication with FTP is only possible for the S7-1500 using the CP 1543-1. The CP can be an FTP server, FTP client or both. FTP clients can also be third-party systems/PCs.

For the FTP server functionality, configure the CP accordingly in STEP 7.

You can use the FTP client functionality to implement, for example, the establishment and termination of an FTP connection, the transfer and deletion of files on the server. For the FTP client functionality, use the FTP_CMD instruction.
Procedure for setting up FTP server functionality

**Requirement:** The FTP server can be reached via the IPv4 network.

1. Configure an S7-1500 automation system with CPU and CP 1543-1 in the device view of the Devices & networks editor of STEP 7.

   At the same time, you need to select the check box "Permit access with PUT/GET communication from remote partner (PLC, HMI, OPC, ...)" in the HW configuration of the S7-1500 CPU under the "Protection" area navigation in the section "Connection mechanisms".

2. Make the following settings in the properties of the CP under "FTP configuration":
   - Select the "Use FTP server for S7 CPU data" check box.
   - Assign the CPU, a data block and a file name under which the DB for FTP will be stored.

3. Download the hardware configuration to the CPU.
Procedure for setting up FTP client functionality

**Requirement:** The FTP server can be reached via the IPv4 network.

1. Configure an S7-1500 automation system with CPU and CP 1543-1 in the device view of the Devices & networks editor of STEP 7.
   
   At the same time, you need to select the check box "Permit access with PUT/GET communication from remote partner (PLC, HMI, OPC, ...)" in the HW configuration of the S7-1500 CPU under the "Protection" area navigation in the section "Connection mechanisms".

2. Call the FTP_CMD instruction in the user program of the CPU.

3. Set the connection parameters for the FTP server in the FTP_CMD instruction.

4. Create a global DB and within this DB a tag of the type FTP_CONNECT_IPV4, FTP_CONNECT_IPV6 or FTP_CONNECT_NAME.

5. Interconnect the tag within the data block with the FTP_CMD instruction.

6. For the connection to the FTP server, specify the following in the DB:
   - The user name, the password and the IP address for the FTP access in the relevant data type (FTP_CONNECT_IPV4, FTP_CONNECT_IPV6 or FTP_CONNECT_NAME)

7. Download the hardware configuration and user program to the CPU.

Application examples

- Application example: FTP communication with S7-1500 and CP 1543-1
  

- Application example: FTP client communication with S7-1200/1500
  

Additional information

The STEP 7 online help describes:

- The system data types
- The instructions for open communication
- The connection parameters
6.10 Establishment and termination of communications relations

Establishment and termination of communications

The table below shows the establishment and termination of communications as part of open communication.

Table 6-5 Establishment and termination of communications

<table>
<thead>
<tr>
<th>Setting up the connection</th>
<th>Establishing communication</th>
<th>Terminating communication</th>
</tr>
</thead>
</table>
| **With the user program** | After downloading the user program to the CPUs:  
The passive communications partner sets up the local connection access by calling TSEND_C/TRCV_C or TCON. Calling TSEND_C/TRCV_C or TCON on the active partner starts connection establishment. If the connection could be established, there is positive feedback to the instructions in the user program.  
After you have terminated a connection using the instruction T_RESET, the connection is reestablished.  
If the connection aborts, the active partner attempts to re-establish the connection. This applies only if the connection was successfully established beforehand with TCON. | • Using the TSEND_C/TRCV_C, TDISCON and T_RESET instructions  
• When the CPU changes from RUN to STOP mode  
• With POWER OFF/POWER ON on a CPU |
| **By configuring a connection** | After downloading the connection configuration and the user program to the CPUs. | By deleting the connection configuration in STEP 7 and downloading the changed configuration to the CPU. |
6.11 **Secure Open User Communication**

### 6.11.1 Secure OUC of an S7-1500 CPU as TLS client to an external PLC (TLS server)

The following section describes how you can set up Open User Communication via TCP from an S7-1500 CPU as TLS client to a TLS server.

**Setting up a secure TCP connection from an S7-1500 CPU as TLS client to a TLS server**

S7-1500 CPUs as of firmware version V2.0 support secure communication with addressing via a Domain Name System (DNS).

For secure TCP communication over the domain name you need to create a data block with the TCON_QDN_SEC system data type yourself, assign parameters and call it directly at one of the instructions TSEND_C, TRCV_C or TCON.

**Requirements:**
- Current date and time are set in the CPU.
- Your network includes at least one DNS server.
- You have configured at least one DNS server for the S7-1500 CPU.
- TLS client and TLS server have all the required certificates.

To set up a secure TCP connection to a TLS server, follow these steps:

1. Create a global data block in the project tree.
2. Define a tag of the data type TCON_QDN_SEC in the global data block.

The example below shows the global data block "Data_block_1" in which the tag "DNS ConnectionSEC" of the data type TCON_QDN_SEC is defined.

<table>
<thead>
<tr>
<th>Data_block_1</th>
<th>Name</th>
<th>Date type</th>
<th>Start value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Static</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DNS Connection SEC</td>
<td>TCON_QDN_SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ConnPara</td>
<td>TCON_QDN</td>
<td></td>
<td>parameter of the TCP connection</td>
</tr>
<tr>
<td>4</td>
<td>InterfaceId</td>
<td>HW_ANY</td>
<td>0</td>
<td>not relevant</td>
</tr>
<tr>
<td>5</td>
<td>ID</td>
<td>CONN_QDN</td>
<td>5</td>
<td>connection reference / identifier</td>
</tr>
<tr>
<td>6</td>
<td>ConnectionType</td>
<td>Byte</td>
<td>11</td>
<td>type of connection, 16#08=11=TCP/IP, 16#13</td>
</tr>
<tr>
<td>7</td>
<td>ActiveEstablished</td>
<td>Bool</td>
<td>true</td>
<td>active/passive connection establishment</td>
</tr>
<tr>
<td>8</td>
<td>RemoteQDN</td>
<td>String[254]</td>
<td>'pic_1.factory127.'</td>
<td>fully or partially qualified domain name of rem</td>
</tr>
<tr>
<td>9</td>
<td>RemotePort</td>
<td>UInt</td>
<td>4000</td>
<td>remote UDP/TCP port number</td>
</tr>
<tr>
<td>10</td>
<td>LocalPort</td>
<td>UInt</td>
<td>0</td>
<td>local UDP/TCP port number</td>
</tr>
<tr>
<td>11</td>
<td>ActivateSecureConn</td>
<td>Bool</td>
<td>true</td>
<td>activate the security functionality of that conn</td>
</tr>
<tr>
<td>12</td>
<td>TLSServerReqClientCert</td>
<td>Bool</td>
<td>false</td>
<td>Just for server side: The TLS server requests a</td>
</tr>
<tr>
<td>13</td>
<td>ExtTLSCapabilities</td>
<td>Word</td>
<td>16#0</td>
<td>Bit 0: Just for client side: validate given IPv4 address</td>
</tr>
<tr>
<td>14</td>
<td>TLSServerCertRef</td>
<td>UDInt</td>
<td>7</td>
<td>for Server side: Reference to own X.509 V3 certificate.</td>
</tr>
<tr>
<td>15</td>
<td>TLSClientCertRef</td>
<td>UDInt</td>
<td>0</td>
<td>for Client side: add id of own X.509 V3 client certificate</td>
</tr>
</tbody>
</table>

Figure 6-12 Data type TCON_QDN_SEC
3. Set the connection parameters of the TCP connection in the "Start value" column. Enter the fully qualified domain name (FQDN) of the TLS server, for example, for "RemoteQDN".

4. Set the parameters for secure communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "ExtTLSCapabilities": If you enter the value 1, the client validates the subjectAlternateName in the X.509-V3 certificate of the server to verify the identity of the server. This validation is executed in the context of the instruction.
   - "TLSServerCertRef": ID of the X.509-V3 certificate (usually a CA certificate) that is 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 certificate store to validate the server authentication.
   - "TLSClientCertRef": ID of the own X.509-V3 certificate.
5. Create one of the instructions TSEND_C, TRCV_C or TCON in the program editor.

6. Interconnect the CONNECT parameter of one of the instructions TSEND_C, TRCV_C or TCON with the tags of the data type TCON_QDN_SEC.

In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag "DNS connectionSEC" (data type TCON_QDN_SEC).

![TCON instruction diagram]

**Figure 6-14  TCON instruction**

**Additional information**

You can find more information on the TCON_QDN_SEC system data type in the STEP 7 online help.

For additional information on secure communication, refer to the section Secure Communication (Page 37).
Secure OUC of an S7-1500 CPU as TLS server to an external PLC (TLS client)

The following section describes how you can set up Open User Communication via TCP from an S7-1500 CPU as TLS server to a TLS client.

Setting up a secure TCP connection via the domain name of the communication partner

S7-1500 CPUs as of firmware version V2.0 support secure communication with addressing via a Domain Name System (DNS).

For secure TCP communication over the domain name you need to create a data block with the TCON_QDN_SEC system data type yourself, assign parameters and call it directly at one of the instructions TSEND_C, TRCV_C or TCON.

Requirements:
- Current date and time are set in the CPU.
- Your network includes at least one DNS server.
- You have configured at least one DNS server for the S7-1500 CPU.
- TLS client and TLS server have all the required certificates.

To set up a secure TCP connection to a TLS client, follow these steps:
1. Create a global data block in the project tree.
2. Define a tag of the data type TCON_QDN_SEC in the global data block.

The example below shows the global data block "Data_block_1" in which the tag "DNS ConnectionSEC" of the data type TCON_FDL_SEC is defined.

<table>
<thead>
<tr>
<th>Data_block_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

Figure 6-15 TCON_QDN_SEC_Server

3. Set the connection parameters of the TCP connection in the "Start value" column. Enter, for example, the local ID of the TCP connection for "ID".
4. Set the parameters for secure communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerReqClientCert": Request for an X.509-V3 certificate from the TLS client.
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.

5. Create one of the instructions TSEND_C, TRCV_C or TCON in the program editor.

6. Interconnect the CONNECT parameter of one of the instructions TSEND_C, TRCV_C or TCON with the tags of the data type TCON_QDN_SEC.

   In the example below, the CONNECT parameter of the TCON instruction is interconnected with the tag "DNS connectionSEC" (data type TCON_QDN_SEC).
Additional information

You can find more information about the system data types TCON_QDN_SEC in the STEP 7 online help.

For additional information on secure communication, refer to the section Secure Communication (Page 37).

6.11.3 Secure OUC between two S7-1500 CPUs

The following section describes how you can set Secure Open User Communication via TCP between two S7-1500 CPUs. In the process one S7-1500 CPU acts as TLS client (active establishing of the connection) and the other S7-1500 CPU as TLS server (passive establishing of the connection).

Setting up a secure TCP connection between two S7-1500 CPUs

For secure TCP communication between two S7-1500 CPUs you need to create a data block with the TCON_IP_V4_SEC system data type yourself in every CPU, assign parameters and call it directly at one of the instructions TSEND_C, TRCV_C or TCON.

Requirements:

- Current date and time are set in the CPU.
- Both S7-1500 CPUs have at least firmware version V2.0
- TLS client and TLS server have all the required certificates.

Figure 6-18 Certificate handling for Secure OUC between two S7-1500 CPUs
Settings at the TLS client
To set up a secure TCP connection in the TLS client, follow these steps:

1. Create a global data block in the project tree.
2. Define a tag of the data type TCON_IP_4_SEC in the global data block.

   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS-Client" of the data type TCON_IP_V4_SEC is defined.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Start value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEC connection 1 TLS-Client</td>
<td>TCON_IP_V4_SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConnPara</td>
<td>TCON_IP_v4</td>
<td></td>
<td>parameter of the TCP connection</td>
</tr>
<tr>
<td>Interfaceld</td>
<td>HW_ANY</td>
<td>72</td>
<td>HW-identifier of IE-interface submodule</td>
</tr>
<tr>
<td>ID</td>
<td>CONN_QUC</td>
<td>10</td>
<td>connection reference / identifier</td>
</tr>
<tr>
<td>ConnectionType</td>
<td>Byte</td>
<td>11</td>
<td>type of connection: 11-TCPIP, 19-UDP (17-TCPIP)</td>
</tr>
<tr>
<td>ActiveEstablished</td>
<td>Bool</td>
<td>true</td>
<td>active/passive connection establishment</td>
</tr>
<tr>
<td>RemoteAddress</td>
<td>IP_V4</td>
<td></td>
<td>remote IPAddress (IPv4)</td>
</tr>
<tr>
<td>ADDR[1]</td>
<td>Byte</td>
<td>192</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>ADDR[2]</td>
<td>Byte</td>
<td>168</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>ADDR[3]</td>
<td>Byte</td>
<td>1</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>ADDR[4]</td>
<td>Byte</td>
<td>100</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>RemotePort</td>
<td>UInt</td>
<td>4711</td>
<td>remote UDP/TCP port number</td>
</tr>
<tr>
<td>LocalPort</td>
<td>UInt</td>
<td>4711</td>
<td>local UDP/TCP port number</td>
</tr>
<tr>
<td>ActivateSecureConn</td>
<td>Bool</td>
<td>true</td>
<td>activate the security functionality of that connection in general</td>
</tr>
<tr>
<td>TLSv3ServerReqClientCert</td>
<td>Bool</td>
<td>false</td>
<td>Just for server side. The TLS server requests a client certificate</td>
</tr>
<tr>
<td>ExtTLSCapabilities</td>
<td>Word</td>
<td>16FO</td>
<td>Bit 0: Just for client side: validate given IPv4 address against the host list; Bit 1: Define caching behavior of the chosen protocol</td>
</tr>
<tr>
<td>TLLServerCertRef</td>
<td>UDInt</td>
<td>1</td>
<td>for Server side: Reference to own X.509 V3 server certificate; for client side: reference to own X.509 V3 client certificate; for further information, see section 6.12.2.5.4.</td>
</tr>
<tr>
<td>TLLClientCertRef</td>
<td>UDInt</td>
<td>5</td>
<td>for Client side: reference to own X.509 V3 client certificate; for further information, see section 6.12.2.5.4.</td>
</tr>
</tbody>
</table>

Figure 6-19 IP_V4_SEC_Client

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS server for "RemoteAddress".

   **Note**

   **Connection parameters Interfaceld and ID**

   Note that you can enter the value "0" for the interface ID and the ID in the data type TMAIL_V4_SEC. In this case the CPU itself searches for a matching local CPU interface or a free connection ID.
4. Set the parameters for secure communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerCertRef": Enter the value 2 (reference to the CA certificate of the TIA Portal project (SHA256) or the value 1 (reference to the CA certificate of the TIA Portal project (SHA1)). If you use a different CA certificate, enter the corresponding ID from the certificate manager of the global security settings.
   - "TLSClientCertRef": ID of the own X.509-V3 certificate.

5. Create one of the instructions TSEND_C, TRCV_C or TCON in the program editor.

6. Interconnect the CONNECT parameter of one of the instructions TSEND_C, TRCV_C or TCON with the tags of the data type TCON_IP_V4_SEC.

**Settings at the TLS server**

To set up a secure TCP connection in the TLS server, follow these steps:

1. Create a global data block in the project tree.

2. Define a tag of the data type TCON_IP_4_SEC in the global data block.

   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS-Server" of the data type TCON_IP_V4_SEC is defined.

   ![Data Block Example](image)

   Figure 6-20  IP_V4_SEC_Server

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS client for "RemoteAddress".
4. Set the parameters for secure communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerReqClientCert": Request for an X.509-V3 certificate from the TLS client. Enter the value "true".
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.
   - "TLSClientCertRef": Enter the value 2 (reference to the CA certificate of the TIA Portal project (SHA256)) or the value 1 (reference to the CA certificate of the TIA Portal project (SHA1)). If you use a different CA certificate, enter the corresponding ID from the certificate manager of the global security settings.

5. Create one of the instructions TSEND_C, TRCV_C or TCON in the program editor.

6. Interconnect the CONNECT parameter of one of the instructions TSEND_C, TRCV_C or TCON with the tags of the data type TCON_IP_V4_SEC.

   In the example below, the CONNECT parameter of the TSEND_C instruction is interconnected with the "SEC connection 1 TLS client" tags (data type TCON_IP_4_SEC).

![Figure 6-21 TSEND_C](image)

**Additional information**

You can find more information about the system data types TCON_IP_4_SEC in the STEP 7 online help.

For additional information on secure communication, refer to the section Secure Communication (Page 37).
6.11.4 Secure OUC via CP interface

The following sections describes the particular points to be taken into consideration in the case of Secure Open User Communication via a CP interface. At least one station is an S7-1500 station with the following modules:

- S7-1500 CPU as of firmware version V2.0 (with the exception of S7-1500 Software Controller)
- CP 1543-1 as of firmware version V2.0 or CP 1543SP-1 as firmware version V1.0

The CP acts in an S7-1500 station as a TLS client (active connection establishment) or a TLS server (passive connection establishment).

The fundamental procedure and the concept for using secure communication via a CP interface is similar to that of secure communication via the interfaces of the S7-1500 CPUs. Essentially, you have to assign the certificates to the CPU in the role of a TLS server or TLS client and not to the CPU. Other rules and procedures therefore apply. These are described below.

Handling certificates for CPs

The following applies in general: You have to be logged on at the certificate manager in the global security settings. The generation of self-signed certificates also requires logon for the global security settings. You have to have sufficient rights as a user (administrator or user with the "Standard" role with the right to "Configure security").

The starting point for the generation or assignment of certificates at the CP is the section "Security > Security properties". In this section, you log on for the global security settings.

Procedure:

1. In the network view of STEP 7, mark the CP and select the section "Security > Security properties" in the Inspector window.
2. Click on the "User logon" button.
3. Log on using your user name and password.
4. Enable the "Activate security functions" option.
   The security properties are initialized.
5. Click in the first line of the "Device certificates" table to generate a new certificate or select an existing device certificate.
6. If the communication partner is also an S7-1500 station, you also have to assign a device certificate to the communication partner with STEP 7 as described here or for the S7-1500 CPU.
Example: Setting up a secure TCP connection between two S7-1500 CPUs via CP interfaces

For secure TCP communication between two S7-1500 CPUs you need to create a data block with the TCON_IPv4_SEC system data type yourself in every CPU, assign parameters and call it directly at the one of the instructions TSEND_C, TRCV_C or TCON.

Requirements:

- Both S7 1500 CPUs have at least firmware version V2.0. If you use the CP 1543SP-1: Firmware version as of V1.0.
- Both CPs (for example CP 1543-1) must have at least firmware version V2.0
- TLS client and TLS server have all the required certificates.
  - A device certificate (end-entity certificate) for the CP must be generated and be located in the certificate memory of the CP. If a communication partner is an external device (for example an MES or ERP system), a device certificate also has to exist for this device.
  - The root certificate (CA certificate) with which the device certificate of the communication partner is signed must also be located in the certificate memory of the CP or in the certificate memory of the external device. If you use intermediate certificates, you have to ensure that the complete certificate path exists in the validating device. A device uses these certificates to validate the device certificate of the communication partner.
- The communication partner must always be addressed via its IPv4 address, not via its domain name.

The following figure shows the different certificates in the devices for the case that both communication partners communicate via a CP 1543-1. In addition, the figure shows the transfer of the device certificates during establishment of the connection ("Hello").

Figure 6-22 Certificate handling in secure OUC between two S7-1500 CPUs via CP interfaces.
Settings at the TLS client

To set up a secure TCP connection in the TLS client, follow these steps:

1. Create a global data block in the project tree.

2. Define a tag of the data type TCON_IP_4_SEC in the global data block. To do so, enter the string "TCON_IP_V4_SEC" in the "Data type" field.

   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS-Client" of the data type TCON_IP_V4_SEC is defined.

   The Interface ID has the value of the HW identifier of the IE interface of the local CP (TLS client).

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS server for "RemoteAddress".

4. Set the parameters for secure communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerCertRef": Enter the value 2 (reference to the CA certificate of the TIA Portal project (SHA256)) or the value 1 (reference to the CA certificate of the TIA Portal project (SHA1)). If you use a different CA certificate, enter the corresponding ID from the certificate manager of the global security settings.
   - "TLSClientCertRef": ID of the own X.509-V3 certificate.
5. Create one of the instructions TSEND_C, TRCV_C or TCON in the program editor.

6. Interconnect the CONNECT parameter of one of the instructions TSEND_C, TRCV_C or TCON with the tags of the data type TCON_IP_V4_SEC.

**Settings at the TLS server**

To set up a secure TCP connection in the TLS server, follow these steps:

1. Create a global data block in the project tree.

2. Define a tag of the data type TCON_IP_4_SEC in the global data block.

   The example below shows the global data block "Data_block_1" in which the tag "SEC connection 1 TLS-Server" of the data type TCON_IP_V4_SEC is defined.

   The interface ID has the value of the HW identifier of the IE interface of the local CP (TLS server).

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Start value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEC connection 1 TLS-Server</td>
<td>TCON_IP_V4_SEC</td>
<td></td>
<td>parameter of the TCP connection</td>
</tr>
<tr>
<td>ConnId</td>
<td>TCON_IP_V4</td>
<td>260</td>
<td>HW-identifier of IE-interface submodule</td>
</tr>
<tr>
<td>InterfaceId</td>
<td>HW_ANY</td>
<td>10</td>
<td>connection reference /identifier</td>
</tr>
<tr>
<td>ConnectionType</td>
<td>Byte</td>
<td>11</td>
<td>type of connection: 11-TCP/IP, 19-UDP (17-TCP/IP)</td>
</tr>
<tr>
<td>ActiveEstablished</td>
<td>Bool</td>
<td>false</td>
<td>active/passive connection establishment</td>
</tr>
<tr>
<td>RemoteAddress</td>
<td>IPv4</td>
<td></td>
<td>remote IP address (IPv4)</td>
</tr>
<tr>
<td>ADDR[1]</td>
<td>Byte</td>
<td>192</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>ADDR[2]</td>
<td>Byte</td>
<td>168</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>ADDR[3]</td>
<td>Byte</td>
<td>1</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>ADDR[4]</td>
<td>Byte</td>
<td>10</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>RemotePort</td>
<td>UInt</td>
<td>4711</td>
<td>remote UDP/TCP port number</td>
</tr>
<tr>
<td>LocalPort</td>
<td>UInt</td>
<td>4711</td>
<td>local UDP/TCP port number</td>
</tr>
<tr>
<td>ActivateSecureConn</td>
<td>Bool</td>
<td>true</td>
<td>activate the security functionality of that connection in general</td>
</tr>
<tr>
<td>TLSServerReqClientCert</td>
<td>Bool</td>
<td>true</td>
<td>Just for server side: The TLS server requests a client certificate</td>
</tr>
<tr>
<td>ExtTLSCapabilities</td>
<td>Word</td>
<td>16#0</td>
<td>Bits: just for client side validate given IPv4 address against the</td>
</tr>
<tr>
<td>TLSServerCertRef</td>
<td>UDInt</td>
<td>6</td>
<td>for Server side: Reference to own X.509 V3 server certificate; for Serv</td>
</tr>
<tr>
<td>TLSClientCertRef</td>
<td>UDInt</td>
<td>1</td>
<td>for Client side: add id of own X.509 V3 client certificate; for Serv</td>
</tr>
</tbody>
</table>

3. Set the connection parameters of the TCP connection in the "Start value" column. For example, enter the IPv4 address of the TLS client for "RemoteAddress".
4. Set the parameters for secure communication in the "Start value" column.
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerReqClientCert": Request for an X.509-V3 certificate from the TLS client. Enter the value "true".
   - "TLSServerCertRef": ID of the own X.509-V3 certificate.
   - "TLSClientCertRef": Enter the value 2 (reference to the CA certificate of the TIA Portal project (SHA256) or the value 1 (reference to the CA certificate of the TIA Portal project (SHA1)). If you use a different CA certificate, enter the corresponding ID from the certificate manager of the global security settings.

5. Create one of the instructions TSEND_C, TRCV_C or TCON in the program editor.

6. Interconnect the CONNECT parameter of the instruction TSEND_C, TRCV_C or TCON with the tags of the data type TCON_IP_V4_SEC.

Upload device as new station

When you upload a configuration with certificates and configured secure Open User Communication as a new station into your STEP 7 project, the certificates of the CP are not uploaded, in contrast to the certificates of the CPU. After the device has been loaded as a new station, no more certificates are contained in the corresponding tables of the CPs for the device certificates.

You have to perform configuration of certificates again after the upload. Otherwise, renewed loading of the configuration results in the certificates that originally exist in the CP being deleted so that secure communication does not function.

Secure OUC connections via CPU and CP interfaces - similarities

- Connection resources:
  No differences between OUC and secure OUC. A programmed secure OUC connection uses a connection resource just like an OUC connection, irrespective of which IE/PROFINET interface communicates with the station.

- Connection diagnostics:
  No differences between OUC and secure OUC connection diagnostics.

- Loading of projects with secure OUC connections into the CPU:
  Only possible in STOP of the CPU, if certificates are loaded as well. Recommendation: Load to device > Hardware and software. Reason: Ensuring the consistency between the program with secure OUC, hardware configuration and certificates.
  Certificates are loaded with the hardware configuration - therefore loading requires a stop of the CPU. The reloading of blocks that utilize further secure OUC connections is only possible in RUN if the certificates required for this purpose are already located on the module.
6.11.5 Secure OUC with Modbus TCP

For secure Modbus TCP connection you need to create a data block with one of the system data types TCON_IP_V4_SEC or TCON_QDN_SEC yourself, assign parameters and call it directly at the MB_Server or MB_CLIENT instruction.

Requirements:

- S7-1500 CPU CPU firmware version V2.5 or higher
- The Modbus client (TLS client) can reach the Modbus server (TLS server) over IP communication in the network.
- TLS client and TLS server have all the required certificates.

Example of setting up a secure Modbus TCP connection to a Modbus TCP server

The following section describes how you can set up a Secure Open User Communication over Modbus TCP from a Modbus TCP client to a Modbus TCP server.

To set up a secure connection from a Modus TCP client (TLS client) to a Modbus TCP server (TLS server) and set up the IPv4 address of the mail server, follow these steps:

1. Create a global data block in the project tree.
2. Define a tag of the data type TCON_IP_V4_SEC in the global data block.

```
<table>
<thead>
<tr>
<th>Data block 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>SEC_ModbusTCP_1</td>
</tr>
<tr>
<td>ConnPara</td>
</tr>
<tr>
<td>InterfaceId</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>ConnectionType</td>
</tr>
<tr>
<td>ActiveEstablished</td>
</tr>
<tr>
<td>RemoteAddress</td>
</tr>
<tr>
<td>ADDR[1]</td>
</tr>
<tr>
<td>ADDR[2]</td>
</tr>
<tr>
<td>ADDR[3]</td>
</tr>
<tr>
<td>ADDR[4]</td>
</tr>
<tr>
<td>RemotePort</td>
</tr>
<tr>
<td>LocalPort</td>
</tr>
<tr>
<td>ActivateSecureConn</td>
</tr>
<tr>
<td>TLSServerReqClientCert</td>
</tr>
<tr>
<td>ExtTLSCapabilities</td>
</tr>
<tr>
<td>TLSClientCertRef</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

3. Set the connection parameters of the TCP connection in the "Start value" column. Enter the IPv4 address of the mail server, for example, for the "MailServerAddress".
4. Set the parameters for secure communication in the "Start value" column. Enter the certificate ID of the CA certificate of the communication partner, for example, for "TLS_ServerCertRef".
   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. In this case you can set up an unsecured Modbus TCP connection.
   - "TLS_ServerCertRef": Reference to the X.509 V3 (CA) certificate of the Modbus TCP server, which is used by the TLS client to validate the authentication of the Modbus TCP server.

5. Create an MB_CLIENT instruction in the program editor.

6. Interconnect the CONNECT parameter of the MB_Client instruction with the tags of the data type TCON_IP_4_SEC.

6.11.6 Secure OUC via e-mail

Setting up a secure connection to a mail server over the CPU interface

For secure communication to a mail server you need to create a data block with one of the system data types TMAIL_V4_SEC, TMAIL_QDN_SEC yourself, assign parameters and call it directly at the TMAIL_C instruction.

Requirements:

- TMAIL_C instruction version V5.0 or higher
- STEP 7 V15 and higher
- S7-1500 CPU V2.5 and higher
- You have assigned all the CA certificates of the mail server (TLS server) to the CPU (TLS client) and have downloaded the configuration to the CPU.
- Current date and time are set in the CPU.
Process for establishing a secure connection to the mail server

You can choose between two processes for establishing the secure connection to the mail server:

- **SMTPS**: The client attempts to immediately establish a TLS connection to the mail server ("handshake" process). If the mail server does not support TLS, then no connection is established.

- **STARTTLS**: Client establishes a TCP connection to the mail server. The client sends a request to "upgrade" the existing connection to a secure TLC connection over the TCP connection. If the mail server supports TLS, the client sends the command to establish a secure connection. The mail server uses the SMTP command "STARTTLS" to do this. The client then establishes a secure connection to the mail server. Advantage: If the mail server does not support TLS, client and mail server can communicate unsecured with each other.

You use the "Remote Port" setting in the data types at the block parameter "MAIL_ADDR_PARAM" to define which process is used for the communication.

<table>
<thead>
<tr>
<th>Process</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTPS:</td>
<td>465&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>STARTTLS</td>
<td>Any (=465)&lt;sup&gt;2&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

<sup>1</sup> The instruction TMAIL_C uses SMTPS only for Port 465. For all other ports STARTTLS is used.

<sup>2</sup> According to RFC, mail servers use Ports 25 and 587 for secure connections with STARTTLS. The use of other port numbers for SMTP is not RFC-compliant, successful communication with such a mail server is not guaranteed.
Example: Setting up a secure connection to a mail server over IPv4

The following section describes how to set up a secure connection to an IPv4 mail server with the TMAIL_C communication instruction.

To set up a secure connection via the IP4 address of the mail server, follow these steps:

1. Create a global data block in the project tree.
2. Define a tag of the data type TMAIL_V4_SEC in the global data block.

The example below shows the global data block "MailConnDB" in which the tag "MailConnectionSEC" of the data type TMAIL_V4_SEC is defined.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datatype</th>
<th>Startwert</th>
<th>Kommentar</th>
</tr>
</thead>
<tbody>
<tr>
<td>MailConnectionSEC</td>
<td>TMAIL_V4_SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InterfaceId</td>
<td>HW, ANY</td>
<td>&quot;Local-CP_1543-1_1-Ethernet-Schnittstelle_1&quot;</td>
<td>Use HW-identifier of the IE-interface to specify the connection reference/identifier</td>
</tr>
<tr>
<td>ID</td>
<td>CONN, LOUC</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>ConnectionType</td>
<td>Byte</td>
<td>16#20</td>
<td>type of connection 16#20-32=TSMail_V4 or TMail_V4</td>
</tr>
<tr>
<td>ActiveEstablished</td>
<td>Bool</td>
<td>true</td>
<td>active/passive connection establishment</td>
</tr>
<tr>
<td>WatchDogTime</td>
<td>Time</td>
<td>TH5000ms</td>
<td>watch dog time to monitor SMTP server association</td>
</tr>
<tr>
<td>MailServerAddress</td>
<td>IP, V4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>Array[1-4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR[1]</td>
<td>Byte</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>ADDR[2]</td>
<td>Byte</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>ADDR[3]</td>
<td>Byte</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ADDR[4]</td>
<td>Byte</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>UserName</td>
<td>String[254]</td>
<td>'myName'</td>
<td></td>
</tr>
<tr>
<td>PassWord</td>
<td>String[254]</td>
<td>'myPW'</td>
<td></td>
</tr>
<tr>
<td>From</td>
<td>EMAIL, ADDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LocalPartPlusAt_</td>
<td>String[64]</td>
<td>'Mustermann@'</td>
<td></td>
</tr>
<tr>
<td>FullQualifiedD...</td>
<td>String[254]</td>
<td>'siemens.com'</td>
<td></td>
</tr>
<tr>
<td>RemotePort</td>
<td>UInt</td>
<td>507</td>
<td></td>
</tr>
<tr>
<td>ActivateSecureComm</td>
<td>Bool</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>ExtTLSCapabilities</td>
<td>Byte</td>
<td>16#0</td>
<td></td>
</tr>
<tr>
<td>ILSServerCertif</td>
<td>UInt</td>
<td>7</td>
<td>Reference to the X.509 V3 (CA) certificate of the server</td>
</tr>
</tbody>
</table>

3. Set the connection parameters of the TCP connection in the "Start value" column. Enter the IPv4 address of the mail server, for example, for the "MailServerAddress".

**Note**

**Connection parameters InterfaceId and ID**

Note that you can enter the value "0" for the interface ID and the ID with instruction version V5.0 or higher of the instruction TMAIL_C in the data type TMAIL_V4_SEC. In this case the CPU itself searches for a matching local CPU interface or a free connection ID.
4. Set the parameters for secure communication in the "Start value" column. Enter the certificate ID of the CA certificate of the communication partner, for example, for "TLSServerCertRef".

   - "ActivateSecureConn": Activation of secure communication for this connection. If this parameter has the value FALSE, the subsequent security parameters are irrelevant. You can set up a non-secure TCP or UDP connection in this case.
   - "TLSServerCertRef": Reference to the X.509 V3 (CA) certificate of the mail server, which is used by the TLS client to validate the authentication of the mail server.

5. Create a TMAIL_C instruction in the program editor.

6. Interconnect the MAIL_ADDR_PARAM parameter of the TMAIL_C instruction with the tag of the data type TMAIL_V4_SEC.

   In the following example the MAIL_ADDR_PARAM parameter of the TMAIL_C instruction is interconnected with the tag "MailConnectionSEC" (data type TMAIL_V4_SEC).

![Figure 6-27 TMAIL_C instruction](image)

### Setting up a secure connection to a mail server over the interface of a communication module

For secure communication to a mail server over a communication module, you need to create a data block with one of the system data types TMAIL_V4_SEC, TMAIL_QDN_SEC or TMAIL_V6_SEC yourself, assign parameters and call it directly at the TMAIL_C instruction.

**Requirements:**
- TMAIL_C instruction with version **V4.0**
- S7-1500 CPU as of firmware version V2.0 with communication module CP 1543-1 as of firmware version V2.0
- ET 200SP CPU as of firmware version V2.0 with communication module CP 1542SP-1 (IRC) as of firmware version V1.0
- You have assigned all the CA certificates of the mail server (TLS server) to the CP (TLS client) and have downloaded the configuration to the CPU.
- Current date and time are set in the CPU.

The STEP 7 online help describes how to set up a secure connection to a mail server over the interface of a communication module.
Application example

This application example (https://support.industry.siemens.com/cs/ww/en/view/46817803) show how you can use the CP of an S7-1500 or S7-1200 station to set up a secure connection to an email server and send an email with the default application "TMAIL_C" from the S7 CPU.

Additional information

You can find more information about the system data types TMail_V4_SEC and TMAIL_QDN_SEC in the STEP 7 online help.

For additional information on secure communication, refer to the section Secure Communication (Page 37).
S7 communication

Characteristics of S7 communication

S7 communication as homogeneous SIMATIC communication is characterized by vendor-specific communication between SIMATIC CPUs (not an open standard). S7 communication is used for migration and for connecting to existing systems (S7-300, S7-400).

For data transfer between two S7-1500 automation systems, we recommend that you use open communication (see section Open User Communication (Page 67)).

Properties of S7 communication

Using S7 communication, the CPU exchanges data with another CPU. Once the user has received the data at the receiver end, the reception data is automatically acknowledged to the sending CPU.

The data is exchanged via configured S7 connections. S7 connections can be configured at one end or at both ends.

S7 communication is possible via:
- Integrated PROFINET or PROFIBUS DP interface of a CPU
- Interface of a CP/CM

S7 connections configured at one end

For an S7 connection configured at one end, the configuration for this connection takes place in only one communication partner and is only downloaded to it.

A one-sided S7 connection can be configured to a CPU that is only a server of an S7 connection (e.g. CPU 315-2 DP). The CPU is configured and the address parameters and interfaces are thus known.

In addition, a one-sided S7 connection can be configured to a partner who is not in the project and whose address parameters and interface and therefore are not known. You need to enter the address; it is not checked by STEP 7. The partner is initially unspecified (no partner address is registered when you create the S7 connection). Once you enter the address, it is "unknown" (i.e. it is named, but the project is unknown).

This makes it possible to use S7 connections beyond the boundaries of a project. The communication partner is unknown to the local project (unspecified) and is configured in another STEP 7 or third-party project.

S7 connections configured at both ends

When an S7 connection is configured at both ends, the configuration and download of the configured S7 connection parameters takes place in both communication partners.
Instructions for S7 communication

For S7 communication with S7-1500, the following instructions can be used:

- **PUT/GET**
  
  You write data to a remote CPU with the PUT instruction. You can use the GET instruction to read data from a remote CPU. The PUT and GET instructions are one-sided instructions, i.e. you need only an instruction in one communication partner. You can easily set up the PUT and GET instructions via the connection configuration.

**Note**

**Data blocks for PUT/GET instructions**

When using the PUT/GET instructions, you can only use data blocks with absolute addressing. Symbolic addressing of data blocks is not possible.

You must also enable this service for protection in the CPU configuration in the "Protection" area.

This FAQ [https://support.industry.siemens.com/cs/ww/en/view/82212115](https://support.industry.siemens.com/cs/ww/en/view/82212115) provides information about how to configure and program an S7 instruction and the GET and PUT communication instructions for data exchange between two S7-1500 CPUs.

- **BSEND/BRCV**
  
  The BSEND instruction sends data to a remote partner instruction of the type BRCV. The BRCV instruction receives data from a remote partner instruction of the type BSEND. You use the S7 communication via the BSEND/BRCV instruction pair for secure transmission of data.

- **USEND/URCV**
  
  The USEND instruction sends data to a remote partner instruction of the type URCV. The URCV instruction receives data from a remote partner instruction of the type USEND. You use the S7 communication via the USEND/URCV instruction pair for fast, non-secure transmission of data regardless of the timing of the processing by the communications partner; for example for operating and maintenance messages.
S7 communication via PROFIBUS DP interface in slave mode

You can find the "Test, commissioning, routing" check box in STEP 7 in the properties of the PROFIBUS DP interface of communications modules (e.g. CM 1542-5). Using this check box, you decide whether the PROFIBUS DP interface of the DP slave is an active or passive device on PROFIBUS.

- Check box selected: The slave is an active device on PROFIBUS.
- Check box cleared: The DP slave is a passive device on PROFIBUS. You can only set up S7 connections configured at one end for this DP slave.

Figure 7-1  "Test, commissioning, routing" check box
Configuring S7 connections for PUT/GET instructions

You can create S7 connections and assign the parameters for these in the connection parameter assignment of the PUT/GET instructions. Changed values are checked immediately by the connection parameter assignment for input errors.

Requirement: A PUT or GET instruction is created in the programming editor.

To configure an S7 connection using PUT/GET instructions, follow these steps:

1. In the program editor, select the call of the PUT or GET instruction.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.

The connection parameters already known are displayed:
- Name of the local end point
- Interface of the local end point
- IPv4 address of the local end point
Figure 7-2  Connection configuration for PUT instruction

4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communication partner.

The following parameters are automatically entered as soon as you have selected the connection partner:

- Name of the partner end point
- Interface of the partner end point. If several interfaces are available, you can change the interface as required.
- Interface type of the partner end point
- Subnet name of both end points
- IPv4 address of the partner end point
- Name of the connection which is used for the communication.

5. If required, change the connection name in the "Connection name" input box. If you want to create a new connection or edit an existing connection, click on the "Select connection" button on the right side next to the input box for the connection name.

Note

The PUT and GET instructions between two communication partners can only run if both the hardware configuration and the program part for the partner end point have been loaded into the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.
Configuring S7 connections for e.g. BSEND/BRCV

If you want to use the instructions for BSEND/BRCV for S7 communication, for example, you first need to configure an S7 connection.

To configure a S7 connection, follow these steps:

2. Select the "Connections" button and the "S7 connection" entry from the drop-down list.
3. Using drag-and-drop, connect the communication partner with each other (via an interface or local end point). If the required S7 subnet does not yet exist, it is created automatically.

   You can also set up a connection to unspecified partners.
4. In the "Connections" tab, select the row of the S7 connection.
5. Set the properties of the S7 connection in the "Properties" tab in the "General" area, for example the name of the connection and the interfaces of the communications partner that will be used.

   For S7 connections to an unspecified partner, set the address of the partner.
   You can find the local ID (reference of the S7 connection in the user program) in the "Local ID" area.
6. In the Project tree, select the "Program blocks" folder for one of the CPUs and open OB1 in the folder by double-clicking on it. The program editor opens.
7. In the program editor, call the relevant instructions for S7 communication in the user program of the communication partner (configured at one end) or in the user programs of the communication partners (configured at both ends). Select the BSEND and BRCV instructions from the "Communication" area of the "Instructions" task card, for example, and drag them to a network of OB1.
8. At the ID parameter of the instruction, assign the local ID of the configured connection to be used for the transmission of data.
9. Assign the parameters for the instructions indicating which data will be written to where and which data will be read from where.
10. Download the hardware configuration and user program to the CPU(s).
S7 communication via CP 1543-1

If you set up S7 communication via the Industrial Ethernet interface of the CP 1543-1, you can select the transport protocol for data transfer in the properties of the S7 connection under "General":

- "TCP/IP" check box selected (default): ISO-on-TCP (RFC 1006): for S7 communication between S7-1500 CPUs
- "TCP/IP" check box cleared: ISO protocol (ISO/IEC 8073): Addressing using MAC addresses

Figure 7-3   Selecting the CP 1543-1 transport protocol
Procedure for setting up an S7 connection via different S7 subnets

You have the option of using an S7 connection over multiple S7 subnets (PROFIBUS, PROFINET/Industrial Ethernet) (S7 routing (Page 260)).

2. Select the "Network" button.
3. Connect the relevant interfaces with the S7 subnets (PROFIBUS, PROFINET/Industrial Ethernet) using drag-and-drop.
4. Select the "Connections" button and the "S7 connection" entry from the drop-down list.
5. Using drag-and-drop in our example, connect PLC_1 in the left S7 subnet (PROFIBUS) to PLC_3 in the right S7 subnet (PROFINET).

The S7 connection between CPU 1 and CPU 3 is configured.

Figure 7-4  S7 connections via different subnets
ET 200SP Open Controller as router for S7 connections

If you assign the "PROFINET onboard [X2]" interface to the CPU 1515SP PC (F) of the SIMATIC PC station, the CPU 1515SP PC (F) can be used as a router for S7 connections. If you use the CP interface for "None, or a different Windows setting", you cannot use the Open Controller as a router for routed S7 connections.

An existing S7 connection routed by the CPU 1515SP PC (F) becomes invalid if the assignment of the interface of the CPU 1515SP PC (F) is changed from "SIMATIC PC station" to "None, or a different Windows setting". Since the PLC now no longer handles routing functions for this connection, when the CPU 1515SP PC (F) is compiled, no message relating to the invalid connection is displayed. The invalid routed S7 connection is displayed only when the end points of the connection are compiled.

The interfaces required for routed S7 connections must remain explicitly assigned on the CPU 1515SP PC (F). You can edit the assignment of the interface of the CPU 1515SP PC (F) in the properties under "PROFINET onboard [X2] > Interface assignment".

Figure 7-5  S7 routing PC station

Additional information

You can find detailed information on configuring S7 connections and how to use the instructions for S7 communication in the user program in the STEP 7 online help.
Point-to-point link

Functionality

A point-to-point connection for S7-1500, ET 200MP and ET 200SP is established via communications modules (CMs) with serial interfaces (RS232, RS422 or RS485):

- **S7-1500/ET 200MP:**
  - CM PtP RS232 BA
  - CM PtP RS422/485 BA
  - CM PtP RS232 HF
  - CM PtP RS422/485 HF

- **ET 200SP:**
  - CM PtP

The bidirectional data exchange via a point-to-point connection works between communications modules or third-party systems or devices capable of communication. At least 2 communication partners are required for communication ("point-to-point"). With RS422 and RS485, more than two communications partners are possible.

Protocols for communication via a point-to-point connection

- Freeport protocol (also called ASCII protocol)
- Procedure 3964(R)
- Modbus protocol in RTU format (RTU: Remote Terminal Unit)
- USS protocol (universal serial interface protocol)

The protocols use different layers according to the ISO/OSI reference model:

- Freeport: Uses layer 1 (physical layer)
- 3964 (R), USS and Modbus: Use layer 1 and 2 (physical layer and data link layer; therefore greater transmission reliability than with Freeport). USS and Modbus use additionally layer 4.

Properties of the Freeport protocol

- The recipient recognizes the end of the data transfer by means of a selectable end criterion (e.g. character delay time elapsed, receipt of end character, receipt of a fixed amount of data).
- The sender cannot recognize whether the sent data arrived free of errors at the recipient.
Properties of procedure 3964 (R)

- When the data is sent, control characters are added (start, end and block check characters). Make sure that these control characters are not included as data in the frame.
- Connection establishment and termination makes use of control characters.
- If transfer errors occur, data transfer is automatically repeated.

Data exchange using Freeport or 3964 (R) communication

The data to be sent is stored in the user program of the corresponding CPU in data blocks (send buffer). A receive buffer is available on the communications module for the received data. Check the properties of the receive buffer and adapt them if necessary. You must create a data block for receiving in the CPU.

In the user program of the CPU, the "Send_P2P" and "Receive_P2P" instructions handle the data transfer between the CPU and CM.

Procedure for setting up Freeport or 3964 (R) communication

1. Configure an S7-1500 configuration with CPU and CM in the device view of the hardware and network editor of STEP 7.
2. Select the interface of the CM in the device view of STEP 7.
3. Assign the parameters of the interface (for example connection communication, configuration of message sending) in the Inspector window of STEP 7 under "Properties" > "General".
4. Select the "Send_P2P" or "Receive_P2P" instruction in the "Instructions" task card under "Communication" > "Communications processor" and drag-and-drop the instruction into the user program (for example into a FB).
5. Assign the parameters for the instructions according to your configuration.
6. Download the hardware configuration and user program to the CPU.

Otherwise: Dynamic parameter assignment of the communications module

In certain types of application it is an advantage to set up communication dynamically; in other words, program-controlled by a specific application.

Typical applications for this, could be, for example manufacturers of serial machines. To make the user interfaces as convenient as possible for their customers, these manufacturers adapt the communications services to the particular operator entries.
Instructions for Freeport communication

There are 3 instructions available for the dynamic configuration in the user program for Freeport communication. The following applies to all 3 instructions: the previously valid configuration data is overwritten but not stored permanently in the target system.

- The "Port_Config" instruction is used for the program-controlled configuration of the relevant port of the communications module.
- The "Send_Config" instruction is used for the dynamic configuration, for example of time intervals and breaks in transmission (serial transmission parameters) for the relevant port.
- The "Receive_Config" instruction is used for dynamic configuration, for example of conditions for the start and end of a message to be transferred (serial receive parameters) for the relevant port.

Instructions for 3964 (R) communication

There are 2 instructions available for dynamic configuration in the user program for 3964 (R) communication. The following applies to the instructions: the previously valid configuration data is overwritten but not stored permanently in the target system.

- The "Port_Config" instruction is used for the program-controlled configuration of the relevant port of the communications module.
- The "P3964_Config" instruction is used for the dynamic configuration of protocol parameters.

Properties of the USS protocol

- Simple, serial data transfer protocol with cyclic message frame traffic in half duplex mode that is tailored to the requirements of drive technology.
- Data transfer works according to the master-slave principle.
  - The master has access to the functions of the drive and can, among other things, control the drive, read status values and read and write the drive parameters.

Data exchange using USS communication

The communications module is the master. The master continuously sends frames (job frames) to the up to 16 drives and expects a response frame from each addressed drive.

A drive sends a response frame under the following conditions:

- When a frame is received without errors
- When the drive is addressed in this frame

A drive must not send if these conditions are not met or the drive was addressed in the broadcast.

The connection to the relevant drives exists for the master once it receives a response frame from the drive after a specified processing time (response delay time).
Procedure for setting up USS communication

1. Configure an S7-1500 configuration with CPU and CM in the device view of the hardware and network editor of STEP 7.

2. In the Project tree, select the "Program blocks" folder and open OB1 in the folder by double-clicking on it. The program editor opens.

3. Select the instructions for USS communication according to your task in the "Communication" area, "Communications processor" folder of the "Instructions" task card and drag them to a network of OB1:
   - The "USS_Port_Scan" instruction allows you to communicate via the USS network.
   - The "USS_Drive_Control" instruction prepares send data for the drive and evaluates the response data of the drive.
   - The "USS_Read_Param" instruction is used to read out parameters from the drive.
   - The "USS_Write_Param" instruction is used to change parameters on the drive.

4. Assign the parameters for the instructions according to your configuration.

5. Download the hardware configuration and user program to the CPU.

Properties of the Modbus protocol (RTU)

- Communication takes the form of serial, asynchronous transfer with a transmission speed of up to 115.2 kbps, half duplex.
- Data transfer works according to the master-slave principle.
- The Modbus master can send jobs for reading and writing operands to the Modbus slave:
  - Reading inputs, timers, counters, outputs, memory bits, data blocks
  - Writing outputs, memory bits, data blocks
- Broadcast to all slaves is possible.

Data exchange using Modbus communication (RTU)

The communications module can be a Modbus master or Modbus slave. A Modbus master can communicate with one or more Modbus slaves (the number depends on the physical interface). Only the Modbus slave explicitly addressed by the Modbus master is permitted to return data to the Modbus master. The slave detects the end of the data transfer and acknowledges it. If an error occurs, it provides an error code to the master.
Procedure for setting up Modbus communication (RTU)

1. Configure an S7-1500 configuration with CPU and CM in the device view of the hardware and network editor of STEP 7.

2. In the Project tree, select the "Program blocks" folder and open OB1 in the folder by double-clicking on it. The program editor opens.

3. Select the instructions for Modbus communication according to your task in the "Communication" area, "Communications processor" folder of the "Instructions" task card and drag them to a network of OB1:
   - The "Modbus_Comm_Load" instruction configures the port of the CM for Modbus communication.
   - The "Modbus_Master" instruction is used for Modbus master functionality.
   - The "Modbus_Slave" instruction is used for Modbus slave functionality.

4. Assign the parameters for the instructions according to your configuration.

5. Download the hardware configuration and user program to the CPU.

Additional information

- You can find more detailed information on communication via point-to-point connections and basics of serial data transmission in the function manual CM PtP communication module - Configurations for point-to-point connections [http://support.automation.siemens.com/WW/view/en/59057093].
- You can find a description of how to use the instructions for point-to-point connections in the user program in the STEP 7 online help.
- You can find information about the communications modules with a serial interface in the manual of the particular communications module.
9.1 What you need to know about OPC UA

9.1.1 OPC UA and Industrie 4.0

Uniform standard for information and data exchange

Industry 4.0 stands for the intensive utilization, evaluation and analysis of the large volumes of data from production in IT systems at the enterprise level. With Industry 4.0, data exchange between the production and enterprise levels is rapidly increasing. However, a prerequisite for success is a uniform standard for the information and data exchange.

The OPC UA (OPC Unified Architecture) standard is particularly suitable for data exchange across different levels thanks to its independence from specific operating systems, its secure transfer procedures and the semantic description of data. OPC UA makes available not only data but also information about the data (data types), at the same time making possible machine-interpretable access to the data.

9.1.2 OPC UA for S7-1500 CPUs

In OPC UA, one system operates as a server and provides data the existing information to other systems (clients).

OPC UA clients, for example, have read and write access to data on an OPC UA server.
OPC UA clients call methods on the OPC UA server.

You can access this data online with a client, including e.g. information on performance and diagnostics. In OPC UA terminology, this function is called "Browsen". The "Subscription" function saves the regular reading of a tag - a client only has values sent to it from the server when a change has occurred.

A system can be both a client and a server.

OPC UA server of the S7-1500 CPU

As of firmware version 2.0, an S7-1500 CPU is equipped with an OPC UA server.

The following sections describe how you configure the OPC UA server of the S7-1500 CPU to make data and methods available for OPC UA clients so that clients have read or write access to PLC tags on the CPU and can call server methods.

The following sections also set out how to integrate companion specifications into the address space of the OPC UA server.
OPC UA client of the S7-1500 CPU

As of firmware version V2.6, an S7-1500 CPU is additionally equipped with an OPC UA client.

The following sections show how to use standardized instructions (PLCopen function blocks) to create a user program that, as an OPC UA client, reads data from an OPC UA server or writes data to an OPC UA server or calls methods from an OPC UA server.

STEP 7 (TIA Portal) assists you in creating user programs by providing an editor for client interfaces and a parameter assignment for OPC UA connections:

The OPC UA instructions for an S7-1500 CPU as client are described in detail in the help to the instructions (Instructions > Communication > OPC UA).

OPC UA client for test purposes

The following description uses various different OPC UA clients to illustrate the use of OPC UA clients:

- "UaExpert" of Unified Automation. An extensive client that can be used free of charge:

- "UA Sample Client" of the OPC Foundation. This client is available free of charge for users who are registered with the OPC Foundation:
  Link for downloading the example client of the OPC Foundation [https://opcfoundation.org]

Application example in Industry online support

Siemens Industry Online Support provides an application example with a client API free of charge. .NET developers can use the functions of this interface to access the OPC UA server of an S7-1500. The client API is based on the .NET OPC UA stack of the OPC Foundation.

The application example shows how to establish connections between servers and clients, for example. It also demonstrates the reading and writing of PLC tags.

9.1.3 General features of OPC UA

The key features of OPC UA

- OPC UA does not depend on a specific operating system platform.
  OPC UA can, for example, be used with Windows, Linux, Mac OS X, a real-time
  operating system or a mobile operating system (such as Android).

- OPC UA is implemented in various different programming languages.
  The OPC Foundation has implemented the OPC UA standard in several programming
  languages: Stacks for ANSI C, .NET and Java are available.

- The OPC Foundation offers the Java stack and .NET stack as well as example programs
  as open source software. See Github [https://github.com/opcfoundation].

- A number of companies offer Software Development Kits (SDK) containing the OPC
  Foundation stacks and other functions to facilitate the development of solutions.

Advantages of using SDKs:
- Support from the supplier
- Tested software
- Detailed documentation
- Clear license conditions (important for selling of solutions)

- Scalability

OPC UA can be used in sensors, embedded systems, controllers, PC systems and
smartphones, as well as in servers running MES and ERP applications.

OPC UA and PROFINET can also be used together. The two protocols use the same
network infrastructure.

- Simple client-server principle

An OPC UA server provides information within a network and an OPC UA client retrieves
this information.

- Integrated security mechanisms

OPC UA uses security mechanisms at various levels:
- A secure connection can only be established between an OPC UA server and an OPC
  UA client if the client and server can register with X.509-v3 certificates and accept
each other's certificates (security at the application level). Various security policies are
possible, including a non-secure connection between server and client (Security
Policy: "No security").
- Before allowing access, a server can request the following information from the user:
  - A certificate (cannot be configured in STEP 7)
  - User name and password
  - No user authorization
The security mechanisms are optional and configurable.

- Independence of a specific transport layer
  The following transport mechanisms are currently supported by OPC UA:
  - The transfer of messages as a binary stream directly via TCP/IP
  - The transfer of messages with XML over TCP/IP and HTTP. This transport mechanism allows only a slow transfer and is therefore almost never used. The S7-1500 CPUs do not support this transport mechanism.
  Binary data exchange is supported by all OPC UA applications (required in OPC UA specification).

- PLC tag mapping
  The information of the OPC UA server (for example the PLC tags) is modeled as nodes connected to one another via references. This makes it possible to browse from node to node with an OPC UA client and find out what content can be read, monitored or written.

- Information
  OPC UA servers provide a lot of information, for example about the CPU, the OPC UA server itself, the data and the data types.

- Instance concept
  OPC UA is based on a type instance concept. Both instances and their type definitions are available in runtime.

- Differentiation of the functionality with profiles: S7-1500 supports the Embedded UA Server Profile, for example

### 9.1.4 From the Classic OPC interface to OPC UA

Standardized interface

Classic OPC only runs on Windows operating systems.

To get around this restriction, the OPC Foundation developed the OPC UA standard.

The standard is not platform-specific and uses an optimized, TCP-based binary protocol for high-performance applications.

Different systems are therefore able to exchange data, for example:

- Controllers with MES and ERP systems
- Siemens controllers with controllers from other manufacturers
- Smartphones with controllers
- Embedded systems with controllers
- Smart sensors with controllers
9.1.5 Addressing nodes

Nodes in the OPC UA address space are uniquely identified by a NodeId (Node ID or Node Identifier).

The NodeId consists of an identifier, identifier type and a namespace index. Namespaces are used to avoid naming conflicts. The OPC Foundation has defined a wide range of nodes that provide information about the given OPC UA server. These nodes can be found in the namespace of the OPC Foundation and have the index 0.

The OPC Foundation also defines data types and tag types.

Namespace

All the tags or methods of an S7-1500 are contained in the namespace (Namespace) "http://www.siemens.com/simatic-s7-opcua". By default this namespace has the Index 3. The index may change later if additional namespaces are inserted into the server or if existing ones are deleted. The current index of the namespace therefore needs to be requested from the server before values are read or written.

The figure below shows the result of such a query. The Siemens "UaClient" program is used as an example.
Identifier

The Identifier corresponds to the name of the PLC tag in quotation marks. The quotation mark is the only sign that is not permitted as part of a name in STEP 7. Quotation marks avoid naming conflicts.

The following example reads the value of the "StartTimer" tag:

<table>
<thead>
<tr>
<th>Index</th>
<th>Boolean Variable</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&quot;StartTimer&quot;</td>
<td>True</td>
</tr>
</tbody>
</table>

The Identifier can consist of several components. The individual components are then separated by a dot. The following example reads the "MyDB" array data block completely. This data block contains an array with ten integer values. All ten values should be read in one pass. Therefore, "0:9" is entered at the array range.

<table>
<thead>
<tr>
<th>Index</th>
<th>Array Data Block of Int16</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&quot;MyDB&quot;:&quot;THIS&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Array Range (for instance 0:9)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7050</td>
</tr>
<tr>
<td>1</td>
<td>7051</td>
</tr>
<tr>
<td>2</td>
<td>7052</td>
</tr>
<tr>
<td>3</td>
<td>7053</td>
</tr>
</tbody>
</table>
9.1 What you need to know about OPC UA

PLC tags in the address space of the OPC UA server

The figure below shows where the PLC tags in the example are located in the address space of the OPC UA server (excerpt from UA client):

The "MyDB" data block is a global data block. The data block is therefore located below the node "DataBlocksGlobal". "StartTimer" is a memory tag and is therefore stored below the "Memory" node.

Figure 9-1  PLC tags in the address space of the OPC UA server
Methods in the address space of the OPC UA server

If you implement a method via your user program, this takes the following form in the address space of the OPC UA Server (see AUTOHOTSPOT):

Figure 9-2 Methods in the address space of the OPC UA server
9.1.6 Mapping of data types

SIMATIC and OPC UA data types

SIMATIC data types do not always correspond with OPC UA data types.

S7-1500 CPUs provide SIMATIC tags (with SIMATIC data types) to their own OPC UA server as OPC UA data types so that OPC UA clients can access these tags over the server interface with OPC UA data types.

A client can read the attribute "DataType" from such a tag and reconstruct the original data type in SIMATIC.

Example

A tag has the SIMATIC data type "COUNTER". You read COUNTER → UInt16 in the table. You now know that you do not need to convert; the COUNTER value is sent over the line as a UInt16 data type.

The client detects from the attribute "DataType" that the tag is actually the SIMATIC data type "COUNTER". With this knowledge, the client reconstructs the data type.

<table>
<thead>
<tr>
<th>SIMATIC data type</th>
<th>OPC UA data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>Boolean</td>
</tr>
<tr>
<td>BYTE</td>
<td>BYTE</td>
</tr>
<tr>
<td></td>
<td>→ Byte</td>
</tr>
<tr>
<td>WORD</td>
<td>WORD</td>
</tr>
<tr>
<td></td>
<td>→ UInt16</td>
</tr>
<tr>
<td>DWORD</td>
<td>DWORD</td>
</tr>
<tr>
<td></td>
<td>→ UInt32</td>
</tr>
<tr>
<td>LWORD</td>
<td>LWORD</td>
</tr>
<tr>
<td></td>
<td>→ UInt64</td>
</tr>
<tr>
<td>SINT</td>
<td>SByte</td>
</tr>
<tr>
<td>INT</td>
<td>Int16</td>
</tr>
<tr>
<td>DINT</td>
<td>Int32</td>
</tr>
<tr>
<td>LINT</td>
<td>Int64</td>
</tr>
<tr>
<td>USINT</td>
<td>Byte</td>
</tr>
<tr>
<td>UINT</td>
<td>UInt16</td>
</tr>
<tr>
<td>UDINT</td>
<td>UInt32</td>
</tr>
<tr>
<td>ULINT</td>
<td>UInt64</td>
</tr>
<tr>
<td>REAL</td>
<td>Float</td>
</tr>
<tr>
<td>LREAL</td>
<td>Double</td>
</tr>
<tr>
<td>S5TIME</td>
<td>S5TIME</td>
</tr>
<tr>
<td></td>
<td>→ UInt16</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
</tr>
<tr>
<td></td>
<td>→ Int32</td>
</tr>
</tbody>
</table>
### OPC UA communication

#### 9.1 What you need to know about OPC UA

<table>
<thead>
<tr>
<th>SIMATIC data type</th>
<th>OPC UA data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTIME</td>
<td>LTIME</td>
<td>Int64</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>UInt16</td>
</tr>
<tr>
<td>TIME_OF_DAY (TOD)</td>
<td>TOD</td>
<td>UInt32</td>
</tr>
<tr>
<td>LTIME_OF_DAY (LTOD)</td>
<td>LTOOD</td>
<td>UInt64</td>
</tr>
<tr>
<td>DATE_AND_TIME (DT)</td>
<td>DT</td>
<td>Byte[8]</td>
</tr>
<tr>
<td>LDT</td>
<td>Date Time</td>
<td></td>
</tr>
<tr>
<td>DTL</td>
<td></td>
<td>mapped as structure</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR</td>
<td>Byte</td>
</tr>
<tr>
<td>WCHAR</td>
<td>WCHAR</td>
<td>UInt16</td>
</tr>
<tr>
<td>STRING (Code page 1252 or Windows-1252)</td>
<td>STRING</td>
<td>String</td>
</tr>
<tr>
<td>WSTRING (UCS-2; Universal Coded Character Set)</td>
<td>WSTRING</td>
<td>String</td>
</tr>
<tr>
<td>TIMER</td>
<td>TIMER</td>
<td>UInt16</td>
</tr>
<tr>
<td>COUNTER</td>
<td>COUNTER</td>
<td>UInt16</td>
</tr>
</tbody>
</table>

**Arrays**

A read or write job with OPC UA is always an array access, which means that it always has an index and length. A single tag is a special case of an array (index 0 and length 1). The data type is simply sent repeatedly on the line. For the tags, the "DataType" attribute indicates the basic data type. The attributes "ValueRank" and "ArrayDimensions" show whether or not you are dealing with an array and how large the array is.
**Data types based on arrays**

There are SIMATIC data types for which an OPC UA value is mapped to an array of bytes. An array of these data types is then mapped to a two-dimensional array.

Example: The SIMATIC data type DATE_AND_TIME (DT) is mapped to an 8-byte array (Byte[8]), see table above. When you define an array of the SIMATIC data type DATE_AND_TIME (DT), it is considered a two-dimensional array.

This fact affects the use of system data types such as OPC-UA_NodeAdditionalInfo and OPC-UA_NodeAdditionalInfoExt, for example:

For the data types described above, you must use the system data type OPC-UA_NodeAdditionalInfoExt for multidimensional arrays instead of OPC-UA_NodeAdditionalInfo.

**Structures**

Structures are transferred as ExtensionObject. The S7-1500 server uses binary representation for transmission of the ExtensionObjects over the line; the individual structure elements come one after the other. At the front is the NodeId of the data type; this is used by the client to establish the structure.

For the OPC UA specification <= V1.03, a client must read, decode and interpret the complete DataTypeDictionary for this purpose (unless it has already done so offline with an XML import).

**Additional information**

More details on mapping of basic data types, arrays and structures can be found in the OPC UA Specification Part 6, "Mappings" (see OPC UA BINARY there).

**9.1.7 What you need to know about OPC UA clients**

**Basics of OPC UA clients**

OPC UA clients are programs that do the following:

- Access the information from an OPC UA server (for example an S7-1500 CPU): read, write, subscriptions
- Execute methods through the OPC UA server

However, OPC US clients can access data that is enabled for this purpose (see "Managing write and read rights").

You need the endpoint of the server to establish a connection to an OPC UA server (see "Endpoints of the OPC UA servers (Page 157)").
Reading out information from the OPC UA server

When a connection to an end point of the server exists, you can use the navigation function of the client: You navigate starting from a defined starting point (from the "root" node) through the address space of the server.

The following information is provided in the process:

- Enabled PLC tags, data blocks and data block components
- Namespace index and identifiers of these PLC tags, data blocks and DB components
- Data types of the PLC tags and DB components
- Number of components in arrays (required for reading and writing arrays)

In addition, you receive information about the OPC UA server itself as well as information about the S7-1500 based on the "OPC UA for Devices" standard of the OPC Foundation (for example, serial number, firmware version).

Reading data from the server and writing to the server

You now know the namespace, identifier and data type of PLC tags. This means that you can now specifically read individual PLC tags and DB components as well as complete arrays and structures. Examples for reading Boolean tags and array data blocks are available under Addressing nodes (Page 132).

Rules for access to structures are available here (Page 243).

With the information that you obtain while navigating through the address space of the server (index, identifier and data type), you can also transfer values to the S7-1500 with the OPC UA client. The following example overwrites the first three values in the array data block "MyDB".

For "Array Range" you specify which components of the array you want to overwrite. The "Good" status code indicates that the values were transferred successfully. However, you can only write the values to the S7-1500 but not the time stamps of these values. The time stamps can only be read.

<table>
<thead>
<tr>
<th>Index</th>
<th>Array Datablock of int16</th>
<th>Values</th>
<th>Status Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&quot;MyDB&quot;+&quot;THIS&quot;</td>
<td>1 2 3</td>
<td>Good</td>
</tr>
</tbody>
</table>

Array Range (for instance 0.9)
0 2
Faster access through registration

The examples up to now use strings as Identifier, for example, "MyBD2"."THIS". If, however, an Identifier is used as a numeric Node ID instead of a String Node ID, access is much faster. If you access specific tags regularly, you should use the functions "RegisteredRead" and "RegisteredWrite":

In this case, your client first logs in the PLC tag on the server. The server returns an Identifier that the client uses for the actual access. This Identifier applies solely to the current session and has to be queried again when the session connection is terminated/lost.

In the following example, the "StartTimer" tag is first registered on the server. Afterwards, the rapid function "RegisteredWrite" is used for setting the value.

<table>
<thead>
<tr>
<th>Index</th>
<th>Boolean Variable</th>
<th>Value</th>
<th>Status Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&quot;StartTimer&quot;</td>
<td>True</td>
<td>Good</td>
</tr>
</tbody>
</table>

In accordance with the same scheme, the "RegisteredRead" function can also be used, which is particularly useful for recurring data readouts. Take into account, however, that depending on the application it may be advisable to use a Subscription instead.

Recommendation: It is best to place registrations in the startup program of the OPC UA client, since the registration takes up time.

Please note that you can set the maximum number of registered nodes in the properties of the S7-1500 CPU and that the Clients have to respect this number, see AUTOHOTSPOT.
## Subscription

The term "Subscription" is used for a function in which only those tags for which an OPC UA client has registered at the OPC UA server are transferred. The OPC UA server only sends a message to the OPC UA client for these registered tags (Subscriptions) when a value has changed. The monitoring of these tags makes constant sampling by the OPC UA client (Polling) superfluous, which reduces the network load.

You have to create a Subscription to use this function. For this purpose, you specify the "Publishing Interval" at the UA client and click the "Create" button. The publishing interval is the time interval in which the server sends new values to the client in a notification (data change notification).

In the following example a subscription has been created: The client receives a message with the new values (publishing interval 50 ms) every 50 milliseconds here.

### Preventing server overload

You can set the OPC UA server of the S7-1500 CPU by means of the "Minimum publishing interval" in such a way that it does not serve extremely short send intervals requested by the client; see Settings of the OPC UA server.

Example: A client wants to be operated at a publishing interval of 50 ms as detailed above. Such a short publishing interval would, however, result in a high network and server load. You should therefore set 1000 ms as the "Minimum publishing interval" for the server. Clients whose subscription requires shorter publishing intervals are "slowed" to 1000 ms and the server is protected from overload.

Sampling and transmission (Sampling & Publishing) within the scope of a subscription are communication processes which, like other communication processes (TCP/UDP/Web server communication...), are processed by the CPU with priority 15. OBs with higher priority interrupt the communication. If you set the sampling and publishing intervals too short, this setting causes a high communication load. Therefore, select intervals as large as possible, which are still sufficient for the application.

For information about the consistency of tags, refer to Consistency of CPU tags (Page 220).
**Monitoring of PLC tags**

When the Subscription has been created, you inform the server which tags are to be monitored with it. In the following example, the "Voltage" tag was added to the subscription.

<table>
<thead>
<tr>
<th>Index</th>
<th>LREAL Variable</th>
<th>Sampling Interval</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&quot;Voltage&quot;</td>
<td>-1</td>
<td>2.21426504</td>
</tr>
</tbody>
</table>

Queue Size | Deadband
---|---
1 | 0.1

The "Voltage" tag contains the value of a voltage that is detected by an S7-1500 CPU.

The sampling interval ("Sampling Interval") contains a negative value (-1). This determines that the default setting of the OPC UA server is used for the sampling interval. The default setting is defined by the "Publishing Interval" of the subscription. If you want to set the smallest possible sampling interval, select the value "0".

In this example, the length of the queue is set to "1": Only one value is read from the CPU at an interval of 50 milliseconds and subsequently sent to the OPC UA client when the value has changed.

The "Deadband" parameter in this example is "0.1": Changes in value have to amount to 0.1 Volt; only then does the sender send the new value to the client. The server does not send smaller changes in value. You can use this parameter, for example, to disable signal noise: Slight changes in a process variable which do not have a real meaning.
9.2 Security at OPC UA

9.2.1 Security settings

Addressing risks

OPC UA allows the exchange of data between different systems, both within the process and production levels and to systems at the control and enterprise level.

This possibility also entails security risks. That is why OPC UA uses a range of security mechanisms:

- Verification of the identity of OPC UA server and clients.
- Checking of the identity of the users.
- Signed/encrypted data exchange between OPC UA server and clients.

These security policies should only be bypassed in cases where it is absolutely necessary:

- During commissioning
- In stand-alone projects without external Ethernet connection

If you have selected the endpoint "None" for "UA Sample Client" of the OPC Foundation, for example, the program issues a clear warning:

![Warning: Selected Endpoint has no security]

When STEP 7 compiles your project it also checks whether you have considered the setting options for the protection and warns you of possible risks. This also includes an OPC UA security policy with the setting "no security", which corresponds to the end point "None".

Note

Disabling security policies you do not want

If you have selected all security policies (default setting) in the secure channel settings of the S7-1500 OPC UA server – in other words the end point "None" (no security) – non-secure data traffic (neither signed nor encrypted) between the server and client is also possible. The OPC UA server of the S7-1500 CPU also sends its public certificate to the client at "None" (No security). And some clients check this certificate. However, the client is not forced to send a certificate to the server. The identity of the client may possibly remain unknown. Each OPC UA client can then connect to the server irrespective of any subsequent security settings.

When configuring the OPC UA server, make sure that only security policies that are compatible with the security concept for your machine or plant are selected. All other security policies should be disabled.

Recommendation: Use the setting "Basic256Sha256 - Sign and Encrypt", which means that the server only accepts Sha256 certificates. The security policy "Basic128Rsa15" is disabled by default and should not be used as an end point. Select end points with a higher security policy.
**OPC UA communication**

**9.2 Security at OPC UA**

### Additional security rules

- Only use the end point "None" in exceptional cases.
- Only use the "guest authentication" of the user in exceptional cases.
- Only allow access to PLC tags and DB components via OPC UA if it is genuinely necessary.
- Use the list of trusted clients in the settings of the S7-1500 OPC UA client to allow access to certain clients only.

### 9.2.2 Certificates pursuant to ITU X.509

Security mechanisms are integrated in several layers in OPC UA. Digital certificates have an important role here. An OPC UA client can only establish a secure connection to an OPC UA server when the server accepts the digital certificate of the client and classifies it as trusted.

See section "Configuring the OPC UA server of the S7-1500 CPU".

The client must also check and trust the certificate of the server. The server and client must show their identities and prove that they are what they claim to be: They must prove their identity. Mutual authentication of client and server, for example, prevents man-in-the-middle attacks.

### Man-in-the-middle attacks

A "man-in-the-middle" could have positioned itself between server and client. A man-in-the-middle is a program that intercepts communication between server and client and claims to be a client or server, and is thus able to obtain information about the S7 program or to set values in the CPU and attack a machine or plant.

OPC UA uses digital certificates that meet standard X.509 of the International Telecommunication Union (ITU).

This allows the identity of a program, a computer or an organization to be proven (authenticated).
X.509 certificates

An X.509 certificate includes the following information:

- Version number of the certificate
- Serial number of the certificate
- Information on the algorithm used by the certificate authority to sign the certificate.
- Name of the certificate authority
- Start and end of the validity period of the certificate
- Name of the program, person or organization for which/whom the certificate has been signed by the certificate authority.
- The public key of the program, person or organization.

An X509 certificate thus links an identity (name of a program, person or an organization) to the public key of the program, person or organization.

Check during connection establishment

When a connection is being established between the client and server, the devices check all information from the certificate that is required to establish integrity, for example signature, validity, application name (URN) and as of firmware V2.5 also the IP addresses of the client in the client certificate.

The validity period stored in the certificate is also checked. The CPU clock must therefore be set and date/time must be within the validity period, otherwise no communication takes place.

Signing and encryption

To allow you to check whether a certificate has been manipulated, certificates are signed. There are various possible procedures here:

- Within the TIA Portal you have the possibility to generate and sign certificates. If you have protected your project and are logged in as a user with the function right to make security settings, you can use the global security settings. The global security settings allow access to the certificate manager and therefore to the certificate authority (CA) of the TIA Portal.

- Additional options are available for creating and signing certificates. In the TIA Portal, you can import certificates into the global certificate manager.
  - You contact a certificate authority (CA) and have your certificate signed.
    In this case, the certificate authority checks your identity and signs your certificate with the private key of the certificate authority. For this purpose you send a CSR (Certificate Signing Request) to the certificate authority. The process of creating a CSSR with the OpenSSL tool is described here. (Page 148)
  - You yourself create a certificate and sign it.
    To this purpose you use, for example, the "Opc.Ua.CertificateGenerator" program of the OPC Foundation. The procedure is described here (Page 37). Or use OpenSSL: Instructions are available under Generating PKI key pairs and certificates yourself (Page 149).
Useful information: Certificate types

- Self-signed certificate:
  Each device generates and signs its own certificate. Application examples: Static configuration with limited number of communication nodes.
  No new certificates can be derived from a self-signed certificate. However, you need to load all self-signed certificates from partner devices to the CPU (STOP required).

- CA certificate:
  All certificates are generated and signed by a certificate authority. Application examples: Dynamically growing plants.
  You only need to download the certificate from the certificate authority to the CPU. The certificate authority can generate new certificates (partner devices can be added without CPU STOP).

Signing

The signature makes it possible to prove the integrity and source of a message as detailed below.

Signing starts with the sender creating a hash value from the plain text (plain text message). The sender then encrypts the hash value with its private key and subsequently transfers the plain text together with the encrypted hash value to the recipient. To verify the signature, the recipient needs the public key of the sender (this is contained in the X509 certificate of the sender). The recipient uses the sender's public key to decrypt the hash value received. The recipient then forms the hash value themselves from the plain text received (the hash process is contained in the sender's certificate). The recipient compares the two hash values:

- If the two hash values are identical, the plain text message has reached the receiver unchanged and has not been manipulated.
- If the two hash values do not match, the plain text message has not reached the receiver unchanged. The plain text message has been manipulated or has been distorted during transfer.

Encryption

Encrypting data prevents unauthorized parties from reading the content. X509 certificates are not encrypted; they are public and can be viewed by anyone.

Encryption involves the sender encrypting the plain text message with the public key of the recipient. To do so, the sender requires the recipient's X509 certificate, as it contains the public key of the recipient. The recipient decrypts the message with their private key. Only the recipient can decrypt the message: They alone hold the private key. The private key must therefore never be disclosed.
Secure channel

OPC UA uses the private and public key of client and server to establish a secure connection, the secure channel. Once the secure connection has been established, the client and server generate an internal key. The internal key is only known to the client and server. The client and server use this internal key to sign and encrypt messages. This symmetric process (a shared key) is much faster than asymmetric processes (private and public key).

9.2.3 Certificates with OPC UA

Usage of X509 certificates with OPC UA

OPC UA uses three types of X.509 certificates for establishing connections from client to server:

- OPC UA application certificates

  Such X.509 certificates identify the software instance, the installation of client or server software. For the "Organization name" attribute, you enter the name of the company that uses the software.

  **Note**
  
  The OPC UA server of the S7-1500 uses application certificates also for the security setting "None" (No security). This ensures compatibility to OPC UA V1.1 and earlier versions.

- OPC UA software certificates

  This X.509 certificate identifies a specific version of the client or server software. These certificates contain attributes that describe which tests this version of the software has passed during certification by the OPC Foundation (or recognized test laboratories). For the "Organization name" attribute, you enter the name of the company that has developed or markets the software.

  **Note**
  
  Software certificates are not supported in STEP 7.

- OPC UA user certificates

  This X.509 certificate identifies the specific user who, for example, accesses process data from the OPC UA server of an S7-1500 CPU. This certificate is not required if the user can authenticate themselves with a password, or if anonymous access is configured.

  **Note**
  
  User certificates are not supported in STEP 7.

These certificates are end-entity certificates: They identify, for example, a person, an organization, a company or an instance (installation) of a software.
9.2.4 Creating self-signed certificates

Using the client's certificate generator

Many OPC UA client applications or SDKs are integrated in a sample application that allows you to generate certificates for the client from this application.

The description for certificate generation can generally be found in the context for describing the OPC UA client application.

Example client from the online support

The OPC UA .NET client for the SIMATIC S7-1500 OPC UA server [https://support.industry.siemens.com/cs/ww/en/view/109737901](https://support.industry.siemens.com/cs/ww/en/view/109737901) creates a self-signed software certificate of the client application in the Windows Certificate Store during the first program start. The documentation on this example describes the procedure for handling these certificates.

Using the certificate generator of the TIA Portal

If you use an OPC UA client that does not generate a client certificate, self-signed certificates can be created with STEP 7.

To do this, follow these steps:

1. In the properties of the CPU, double-click "<Add new>" under "Protection & Security > Certificate manager > Device certificates".
2. Click "Add".
3. In the "Create a new certificate" dialog, select the "OPC UA client" option for "Usage".
4. Click "OK".

In the field "Subject Alternative Name" STEP 7 automatically enters the URI for the generated certificate. In the program-specific certificate generation by means of the .NET stack of the OPC Foundation, the field is called, for example, "ApplicationUri" - it can have a different name in other tools for certificate generation.
9.2.5 Generating PKI key pairs and certificates yourself

This section is only relevant if you want to use an OPC UA client that cannot itself create a PKI key pair and a client certificate. In this case, you generate a private and a public key using OpenSSL, generate an X.509 certificate, and sign the certificate yourself.

Using OpenSSL

OpenSSL is a tool for Transport Layer Security that you can use to create certificates. You can also use other tools, for example XCA, a type of key management software with a graphical user interface for an improved overview of certificates issued.

To work with OpenSSL under Windows, follow these steps:

1. Install OpenSSL under Windows. If you are using a 64-bit version of the operating system, install OpenSSL in the "C:\OpenSSL-Win64" directory, for example. You can obtain OpenSSL-Win64 as a download from various providers for open source software.

2. Create a directory, for example "C:\demo".

3. Open the command line (cmd.exe). To do so, click "Start" and enter "cmd" in the search field. Right-click "cmd.exe" in the results list and run the program as an administrator. Windows then opens the command line (DOS prompt)

4. Change to the "C:\demo" directory. To do this, enter the following command: "cd C:\demo".

5. Set the following network variables:
   - `set RANDFILE=c:\demo\.rnd`
   - `set OPENSSL_CONF=C:\OpenSSL-Win64\bin\openssl.cfg`

   The figure below shows the command line with the following commands:

   ![Command Line](image1.png)

6. Now start OpenSSL. If OpenSSL has been installed in the C:\OpenSSL-Win64 directory, enter the following: C:\OpenSSL-Win64\bin\openssl.exe The figure below shows the command line with the following command:

   ![Command Line](image2.png)
7. Generate a private key. Save the key to the "myKey.key" file. The key in this example is
1024 bits long; for greater RSA security, use 2048 bits in practice. Enter the following
command: "genrsa -out myKey.key 2048" ("genrsa -out myKey.key 1024" in the
example). The figure below shows the command line with the command and the output of
OpenSSL:

```
C:\WIND\system32\cmd.exe C:\openssl-Win64\bin\openssl.exe
C:\demo>C:\openssl-Win64\bin\openssl.exe
OpenSSL> genrsa -out myKey.key 2048
Generating RSA private key, 2048 bit long modulus
...*****
+6837 (0x10001)
OpenSSL
```

8. Generate a CSR (Certificate Signing Request). To do this, enter the following command:
"req -new -key myKey.key -out myRequest.csr". During execution of this command,
OpenSSL queries information about your certificate:
- Country name: for example "DE" for Germany, "FR" for France
- State or province name: for example "Bavaria"
- Location Name: for example "Augsburg"
- Organization Name: Enter the name of your company.
- Organizational Unit Name: for example "IT"
- Common Name: for example "OPC UA client of machine A"
- Email Address:

**Important**: Both the IP address and the URL of the client program (application) have to be
stored in the "Subject Alternative Name" field of the created certificate; otherwise, the CPU
will not accept the certificate.

The information you enter is added to the certificate. The figure below shows the command
line with the command and the output of OpenSSL:

```
C:\WIND\system32\cmd.exe C:\openssl-Win64\bin\openssl.exe
OpenSSL> req -new -key myKey.key -out myRequest.csr
You are about to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value.
If you enter '.', the field will be left blank.
Country Name [2 letter code] [AU]:
```

The command creates a file in the C:\demo directory containing the Certificate Signing
Request (CSR); in the example, this is "myRequest.csr".
Using the CSR

There are two ways to use a CSR:

- You send the CSR to a certificate authority (CA): Read the information of the respective certification authority. The certificate authority (CA) checks your information and identity (authentication) and signs the certificate with the private key of the certificate authority. You receive the signed X.509 certificate and use this certificate for OPC UA, HTTPS or Secure OUC (secure open user communication), for example. Your communication partners use the public key of the certificate authority to check whether your certificate was really issued and signed by that CA. The certificate authority has confirmed your information in the certificate.

- You sign the CSR yourself: Using your private key. This option is shown in the next step.

Signing the certificate yourself

Enter the following command so that you can generate and sign your certificate (self-signed certificate) yourself: "x509 -req -days 365 -in myRequest.csr -signkey myKey.key -out myCertificate.crt".

The figure below shows the command line with the command and OpenSSL:

![Command line with OpenSSL](image)

The command generates an X.509 certificate with the attributes that you transfer with the CSR (in the example "myRequest.csr"), for example with a validity of one year (-days 365). The command also signs the certificate with your private key ("myKey.key" in the example). Your communication partners can use your public key (contained in your certificate) to check that you are in possession of the private key that belongs to this public key. This also prevents your public key from being misused by an attacker.

With self-signed certificates, you yourself confirm that the information in your certificate is correct. There is no independent body that checks your information.
9.2.6 Secure transfer of messages

Establishing secure connections with OPC UA

OPC UA uses secure connections between client and server. OPC UA checks the identity of the communication partners. OPC UA uses certificates in accordance with X.509-V3 from the ITU (International Telecommunication Union) for client and server authentication. Exception: A secure connection is not established with the "No security" security policy.

Message security mode

OPC UA uses the following security policies to protect messages:

- **No security**
  All messages are unsecured. In order to use this security policy, establish a connection to a None end point of a server.

- **Signing**
  All messages are signed. This allows the integrity of the messages received to be checked. Manipulations are detected. In order to use this security policy, establish a connection to a Sign end point of a server.

- **Sign & Encrypt**
  All messages are signed and encrypted. This allows the integrity of the messages received to be checked. Manipulations are detected. What is more, no attacker can read the contents of the message (protection of confidentiality). In order to use this security policy, establish a connection to a "SignAndEncrypt" end point of a server.

The security policies are also named according to the algorithms used. Example: "Basic256Sha256 - Sign & Encrypt" means: Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
Layers required

The figure below shows the three layers that are always required for establishing a connection: the transport layer, the secure channel and the session.

- **Transport layer:**
  This layer sends and receives messages. OPC UA uses an optimized TCP-based binary protocol here. The transport layer is the basis for the subsequent secure channel.

- **Secure channel**
  The secure channel receives the data received from the transport layer, and forwards that data to the session. The secure channel forwards data of the session that is to be sent to the transport layer.

  In "Sign" security mode, the secure channel signs the data (messages) that is sent. When a message is received, the secure channel checks the signature to detect any manipulations.

  With a "SignAndEncrypt" security policy, the secure channel signs and encrypts the send data. Data received is decrypted by the secure channel, and the secure channel then checks the signature.

  With the "No security" security policy, the message packages pass the secure channel unchanged (the messages are received and sent in plain text).

- **Session**
  The session forwards the messages from the secure channel to the application, or receives from the application the messages that are to be sent. The application uses the process values or provides the values.
Establishing the secure channel

The secure channel is established as follows:

1. The server starts establishing the secure channel when it receives a request to this effect from the client. This request is signed or signed and encrypted, or the message is sent in plain text (security mode of the selected server end point). With "Sign" and "Sign & Encrypt", the client sends a "secret" (random number) with the request.

2. The server validates the client certificate (contained in the request, unencrypted) and checks the identity of the client. If the server trusts the client certificate,
   - it decrypts the message and checks the signature ("Sign & Encrypt"),
   - checks the signature only ("Sign"),
   - or leaves the message unchanged ("No security")

3. The server then sends a response to the client (same level of security as the request). The server secret is contained in the response. The client and server calculate a symmetric key from the client and server secret. The secure channel is now established.

   The symmetric key (instead of the private and public key of client and server) is now used for signing and encrypting messages.

Establishment of the session

The session is executed as follows:

1. The client starts establishing the session by sending a CreateSessionRequest to the server. This message contains a Nonce, a random number that is only used once. The server must sign this random number (Nonce) to prove that it is the owner of the private key. The private key belongs to the certificate that the server uses to establish the secure channel. This message (and all subsequent messages) is secured in line with the security policies for the selected server endpoint (selected security policies).

2. The server responds with the CreateSession Response. This message contains the public key of the server and the signed Nonce. The client checks the signed Nonce.

3. If the server passes the test, the client sends a SessionActivateRequest to the server. This message contains the information that is required for user authentication:
   - Either the user name and password
   - Or the X.509 certificate of the user (not supported in STEP 7 V15)
   - Or no data (if anonymous access is configured).

4. If the user has the necessary rights, the server returns a message to the client (ActivateSessionResponse). This activates the session.

   The secure connection between the OPC UA client and server has been established.

Establishing a connection to PLCopen function block

The PLCopen specification defines a range of IEC 61131 function blocks for OPC UA clients. The instruction UA_Connect initiates both a secure channel and a session following the pattern described above.
9.3 Using the S7-1500 as an OPC UA server

9.3.1 Useful information about the S7-1500 CPU OPC UA server

9.3.1.1 The OPC UA server of the S7-1500 CPUs

The S7-1500 CPUs as of firmware V2.0 are equipped with an OPC UA server. Apart from the Standard-S7-1500 CPUs this applies to the variants S7-1500F, S7-1500T, S7-1500C, S7-1500pro CPUs, ET 200SP CPUs, SIMATIC S7-1500 SW controllers and PLCSIM Advanced.

Convention: "S7-1500 CPUs" also includes the above-mentioned CPU variants.

S7-1500 CPU OPC UA server basics

Access to the OPC UA server of the CPU is possible via all integrated PROFINET interfaces of the S7-1500 CPU.

Direct access to the OPC UA server of the CPU over the backplane bus of the automation system is not possible via a CP or CM.

For access by clients, the server saves the enabled PLC tags and other information in the form of nodes (see Configuring access to PLC tags). These nodes are interconnected and form a network. OPC UA defines access points to this network (well-known nodes) that enable navigation to subordinate nodes.

With an OPC UA client you can read, observe or write values of tags of the PLC program as well as call methods that are available to the server. As of firmware version 2.5 you can implement methods, see AUTOHOTSPOT.
Node classes

OPC UA servers provide information in the form of nodes. A node can be, for example, an object, a tag, a method or a property.

The example below shows the address space of the OPC UA server of an S7-1500 CPU (extract from the OPC UA client "UaExpert" from Unified Automation).

![Figure 9-4 Example of the address space of the OPC UA server of an S7-1500 CPU](image)

In the figure above, the "MyValue" tag is selected (highlighted in gray). This tag is located below the "Memory" node, which has the node class "Object". "Memory" is below the "PLC_1" node (also an Object).

Address space

The nodes are linked over references, for example, the reference "HasComponent, which represents a hierarchical relationship between a node and its subordinate nodes. With their references, the nodes form a network that can, for example, take the form of a tree. A network of nodes is also called an address space. Starting from the root, all nodes can be reached in the address space.
9.3.1.2 **End points of the OPC UA server**

The end points of the OPC UA server define the security level for a connection. Depending on the purpose of use or desired security level, you have to carry out the corresponding settings for the connection at the end point.

**Different security settings**

Before establishing a secure connection, OPC UA clients ask the server with which security settings connections are possible. The server returns a list with all the security settings (endpoints) that the server offers.

**Structure of end points**

End points consist of the following components:

- Identifier for OPC: "opc.tcp"
- IP address: 192.168.178.151 (in the example)
- Port number for OPC UA: 4840 (standard port)
  
  The port number can be configured, see "Settings of the OPC UA server".

- Security setting for messages (Message Security Mode): None, Sign, SignAndEncrypt.
- Encryption and hash procedures (Security Policy): None, Basic128Rsa15, Basic256, Basic256Sha256 (in the example).

The following figure shows the "UA Sample Client" of the OPC Foundation.

The client has established a secure connection to the OPC UA server of an S7-1500 CPU to the end point "opc.tcp://192.168.178.151:4840 - [SignAndEncrypt: Basic128Rsa15:Binary]". The security settings "SignAndEncrypt:Basic128Rsa15" are contained in the end point.

**Note**

**Select an endpoint with as strict as possible a security policy**

Select an application-appropriate security policy for the end point and disable the less strict security policy at the OPC UA server.

A Sha256 certificate is required for the most secure end points (Basic256Sha256) of the S7-1500 CPU OPC UA server.
9.3 Using the S7-1500 as an OPC UA server

A connection to a server end point is only established if the OPC UA client complies with the security policies of that end point.

Through the information provided by the OPC UA server

OPC UA servers provide a wide range of information:

- The values of PLC tabs and DB components which clients may access.
- The data types of these PLC tags and DB components.
- Information on the OPC UA server itself and on the CPU.

This gives clients an overview and allows them to read out specific information. Previous knowledge of the PLC program and the CPU data is not required. You do not need to ask the developer of the PLC program when PLC tags are to be read. All necessary information is stored on the server itself (for example, the data types of the PLC tags).
Display of the information of the OPC UA server

You have the following options:

- **Online**: You have all the available information displayed during the runtime of the OPC UA server. To do so, navigate (browse) the address space of the server.
- **Offline**: You export an XML file that is based on the XML schemes of the OPC Foundation.

Server methods created by the user (FB instance that can be called by an OPC UA client) are not exported as of STEP 7 V15.1, see AUTOHOTSPOT.

- **Offline with the Openness API**: In your program, you use the API (Application Programming Interface) of the TIA Portal to access the function for exporting all PLC tags that can be read by OPC UA. This requires .NET Framework 4.0; see TIA Portal Openness, Automating SIMATIC projects with scripts [https://support.industry.siemens.com/cs/ww/en/view/109477163](https://support.industry.siemens.com/cs/ww/en/view/109477163).

- **If you already know the syntax and the PLC program, you can access the OPC UA server without first researching the information.**

9.3.1.3 Runtime behavior of the OPC UA server

**OPC UA server in operation**

The OPC UA server of the S7-1500 CPU starts when you activate the server and download the project to the CPU.

How to activate the OPC UA server is described here.

**Response to CPU STOP**

An activated OPC UA server remains in operation even if the CPU switches to "STOP". The OPC UA server continues to respond to requests from OPC UA clients.

Server response in detail:

- **If you request the values of PLC tags, you will get what were the latest values before the CPU switched to or was set to "STOP".**

- **If you write values to the OPC UA server, the OPC UA server will accept those values. However, the CPU will not process the values because the user program is not executed in "STOP" mode.**

An OPC UA client can nonetheless read the values written at STOP from the OPC UA server of the CPU.

During restart, the CPU overwrites the values written at STOP with the start of the PLC tags.

- **If you call a server method, the error message 16#00AF_0000 (BadInvalidState) is output because the server method (user program) is not running.**
9.3 Using the S7-1500 as an OPC UA server

Server restart

The OPC UA server is stopped upon each download to the CPU (for example after a configuration or block is downloaded) and then restarts. The duration of the restart depends on the scope of the data structure.

Reading CPU operating mode over OPC UA server

The OPC UA server allows you to read out the CPU mode, see figure below:

In addition to the operating mode of the CPU you can, for example, read out the diagnostics status of the CPU (Good).
9.3.1.4 Diagnostics of the OPC UA server

Online diagnostics of the OPC UA server

The S7-1500 CPU OPC UA server can be diagnosed online with standard OPC UA clients, such as UaExpert.

The diagnostic information is subdivided into the following areas:

- Server Diagnostics
- Sessions Diagnostics
- Subscriptions Diagnostics

In the address space of the server, for example, the following nodes are available with diagnostic information:

- **ServerDiagnosticsSummary**: Server diagnostics summary
  - **CurrentSessionCount**: Number of active sessions

- **SessionsDiagnosticsSummary**: Session diagnostics summary
  - **SessionTimeout**: Set time that a session lasts, e.g. in the event of disconnection
  - **SecurityRejectedSessionCount**: Number of sessions rejected due to mismatching end point security settings between client and server

- **SubscriptionsDiagnosticsArray**: Array with one element per subscription for each session

![Figure 9-7 Server Diagnostics](image-url)
The SessionsDiagnosticsSummary node also shows the properties of the client application accessing the server within the session.

Diagnostics of the connection between client and server

To diagnose the status of the connection during program runtime in the client, use the following instruction:

OPC_UA_ConnectionGetStatus: Read connection status.
9.3.2 Configuring access to PLC tags

9.3.2.1 Managing write and read rights

Enabling PLC tags and DB tags for OPC UA

OPC UA clients can have read and write access to PLC tags and DB tags if the tags are enabled for OPC UA (default setting). For an enabled tag the check box "Accessible from HMI/OPC UA" is activated.

The following example shows an array data block:

![Figure 9-9 Enabling PLC tags and DB tags for OPC UA](image)

This array can be read completely in one pass by OPC UA clients (see Addressing nodes). The check boxes at "Accessible from HMI/OPC UA" and "Writable from HMI/OPC UA" are activated for all the components of the array.

Result: OPC UA clients can both read and write these components.

Removing write rights

If you want to write-protect a tag, deselect the "Writable from HMI/OPC UA" option for that tag. This removes the write right for the OPC UA clients and HMI devices.

Result: Only read access by OPC UA clients and HMI devices is possible. OPC UA clients cannot assign values to this tag and therefore cannot influence execution of the S7 program.

Removing write and read rights

To write-protect and read-protect a tag, disable the "Accessible from HMI/OPC UA" option for that tag (checkbox not selected). This makes the OPC UA server remove the tag from its address space. OPC UA clients can no longer see that CPU tag.

Result: OPC UA clients and HMI devices can neither read nor write the tag.
Write and read rights of structures

If you remove the read or write right for the component of a structure, the structure or the data block cannot be written or read as a whole.

If you remove read and write rights for individual components of a PLC data type (UDT), the rights will also be removed from any data block based on the UDT.

Visible in HMI engineering

The option "Visible in HMI Engineering" applies to Siemens engineering tools. If you disable the option "Visible in HMI Engineering" (check mark not set), you can no longer configure the tag in WinCC (TIA Portal).

The option does not have any effect on OPC UA.

Rules

- Only allow read access to PLC tags and tags of data blocks in STEP 7 if this is necessary for communication with other systems (controllers, embedded systems or MES).
  
  You should not enable other PLC tags.

- Only allow write access over OPC UA if write rights are genuinely necessary for specific PLC tags and tags of data blocks.

- If you have reset the "Accessible from HMI/OPC UA" option for all elements of a data block, the data block for an OPC UA client is no longer visible in the address space of the OPC UA server of the S7-1500 CPU.

- You can also prevent access to an entire data block centrally (see AUTOHOTSPOT). This setting "overrules" the settings for the components in the DB editor.
9.3.2.2 Managing write and read rights for a complete DB

Hiding DBs or DB contents for OPC UA clients

As of STEP 7 V15, you have the option of easily preventing access to a complete data block by an OPC UA client.

With this option, the data of the corresponding DB, including instance DBs of function blocks, remains hidden for OPC UA clients.

In the default setting, data blocks can be read and written from OPC UA clients.

Procedure

Proceed as follows to completely hide a data block for OPC UA clients or to protect a data block from write access from OPC UA clients:

1. Select the data block to be protected in the project tree.
2. Select the "Properties" shortcut menu.
3. Select the "Attributes" area.
4. Select/clear the "DB accessible from OPC UA" checkbox as required.

Figure 9-10 Hiding DBs or DB contents for OPC UA clients

Note

Effect on settings in the DB editor

If you hide a DB using the DB attribute described here, the settings for the components in the DB editor are no longer relevant; individual components can no longer be accessed or written.
9.3 Using the S7-1500 as an OPC UA server

9.3.2.3 Accessing OPC UA server data

High performance in line with application

OPC UA is designed for the transfer of a high volume of data within a short period of time. You can increase the performance significantly if you do not access individual PLC tags, but rather read and write arrays and structures as a whole.

It is fastest to access arrays. Therefore, you should combine the data for OPC UA clients into arrays.

Recommendations for access to the OPC UA server by the OPC UA client

- For one-off or infrequent data access, use standard read/write access.
- For cyclic access to small amounts of data (up to ca. every 5 seconds), use subscriptions. Optimize the settings for the smallest publishing interval and the smallest sampling interval at the OPC UA server.
- If you access specific tags regularly (recurring access), you should use the functions "RegisteredRead" and "RegisteredWrite".

Allow a greater communication load for the PLC by increasing the value for "Cycle load due to communication". Make sure that your application still works properly with the changed settings.

Procedure for creating an array DB

You can create arrays for example in global data blocks, in the instance data block of a function block or as an array DB. The following sections describe how to create an Array-DB.

To create a data block with an array (array data block), follow these steps:

1. Select the CPU with the OPC UA server in the project tree.
2. Double-click "Program blocks".
3. Double-click "Add new block".
4. Click "Data block".
5. Select a unique name for the data block and accept the name that is already entered.
6. Select the "Array DB" entry from the drop-down list for "Type".
7. Select the data type for the individual components of the array from the drop-down list for "Array data type".
8. Enter the high limit of the array for "Array limit".
9. Click "OK".
9.3.2.4 Export OPC UA XML file

Generating an OPC UA export file

OPC UA specifies an information model XML scheme with a standard syntax, with which manufacturers can define their information model of a system component, for example, and make it available in machine-readable form. A file that uses this scheme is also known as a node set file.

With STEP 7 (TIA Portal) you can easily export the standard SIMATIC information model of the S7-1500 CPU as a server to an OPC UA XML file (node set file); including all PLC variables and methods you have enabled for OPC UA.

You use the OPC UA XML file for the offline configuration of an OPC UA client; it is structured according to the OPC UA specification and acts as a standard SIMATIC server interface.

To create and export the OPC UA XML file, follow these steps:

1. Select the CPU. Click on the CPU symbol (for example in the network view).
2. Click "General > OPC UA > Server > Export" in the properties of the CPU.
3. Click "Export OPC UA XML file".
4. Select the directory in which you want to save the export file.
5. Select a new name for the file or keep the name that is already entered.
6. Click "Save".

**Note**

As of STEP 7 (TIA Portal) V15.1, server methods are contained in the OPC UA export file (node set) together with their input and output parameters.

Exporting all array elements separately

If the "Export all array elements as separate nodes" option is selected in the CPU properties under "OPC UA > Server > Export", the OPC UA XML file contains all elements of arrays as individual XML elements. In addition, the arrays themselves are each described in an XML element in the XML file.

If an array contains many array elements, the XML file can be very large.

**Tip**

The following FAQ contains a converter with which you can convert the export file into CSV format. You then obtain a list of the tags of the CPU that can be accessed by OPC UA.

9.3.3 Configuring the OPC UA server of the S7-1500 CPU

9.3.3.1 Enabling the OPC UA server

Requirement

- You have acquired a Runtime license for the operation of the OPC UA server; see Licenses for the OPC UA server (Page 192).
- When using certificates for secure communication (for example, HTTPS, secure OUC, OPC UA), make sure that the corresponding modules have the current time of day and the current date. Otherwise, the modules will evaluate the certificates used as invalid and secure communication will not work.

Commissioning an OPC UA server

By default, the OPC UA server of the CPU is not enabled for reasons of security: OPC UA clients have neither write nor read access to the S7-1500 CPU.

Follow these steps to activate the OPC UA server of the CPU:

1. Select the CPU. Click on the CPU symbol (for example in the network view).
2. Click "OPC UA > Server" in the properties of the CPU.
3. Activate the OPC UA server of the CPU.
4. Confirm the security notes.
5. Go to the CPU properties, select "Runtime licenses" and set the runtime license acquired for the OPC UA server.
6. Compile the project.
7. Download the project to the CPU.

The OPC UA server of CPU now starts.

Settings remain stored

If you have already enabled the server and made settings, those settings are not lost if the server is disabled. The settings are saved as before and are available when you enable the server again.
Application name

The application name is the name of the OPC UA application. The name is displayed under "OPC UA > General":

- The default setting for the server name is: "SIMATIC.S7-1500.OPC-UAServer:PLC1".
- The default consists of "SIMATIC.S7-1500.OPC-UAServer:" and the name of the CPU selected under "General > Product information > Name", in this case "PLC_1".
- Clients identify the server using the application name.

The example below originates from UaExpert:

If you have activated the server, you can also use a different name that is meaningful in your project and which fulfills the requirements of your project, e.g. for worldwide uniqueness.

Changing the application name

To change the name of the OPC UA server, follow these steps:

1. Select the CPU. Click on the CPU symbol (for example in the network view).
2. Click "OPC UA > General" in the properties of the CPU.
3. Enter a meaningful name.

Please note that the application name is also entered on the certificate (Subject Alternative Name) and you may have to generate an existing certificate again after changing the application name.
9.3.3.2 Access to the OPC UA server

Server addresses

The OPC UA server of the S7-1500 CPU can be accessed over all internal PROFINET interfaces of the CPU (as of firmware V2.0), but not over the PROFINET interfaces of CP/CM.

With SIMATIC S7-1500 SW controllers, access to the OPC UA server is only possible via PROFINET interfaces that are assigned to the software PLC.

In the example, connections to the OPC UA server of the CPU can be established over the following URLs (Uniform Resource Locator):

```
<table>
<thead>
<tr>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>opc.tcp://192.168.178.151:4840</td>
<td>IP address at which the OPC UA server can be accessed from the Ethernet subnet 192.168.178.</td>
</tr>
<tr>
<td>opc.tcp://192.168.1.1:4840</td>
<td>IP address at which the OPC UA server can be accessed from the Ethernet subnet 192.168.1.</td>
</tr>
</tbody>
</table>
```

The URLs are structured as follows:

- Protocol identifier "opc.tcp://"
- IP address
  - 192.168.178.151
    The IP address at which the OPC UA server can be accessed from the Ethernet subnet 192.168.178.
  - 192.168.1.1
    The IP address at which the OPC UA server can be accessed from the Ethernet subnet 192.168.1.
- TCP Port number
  - Default: 4840 (standard port)
    The port number can be changed under "OPC > UA > Server > Port".
Dynamic IP addresses

In the example below, the IP address for the PROFINET interface [X2] has not yet been specified.

The placeholder "<dynamically>" appears in the table.

The IP address of this PROFINET interface is set later on the device, e.g. via the display of the CPU.

Activating the standard SIMATIC Server interface

If the "Activate Standard SIMATIC server interface" option is selected, the OPC UA server of the CPU provides the enabled PLC tags and server methods to the clients, as is specified in the OPC UA specification.

This option is selected in the default setting.

Leave the option selected so that OPC UA clients can automatically connect to the OPC UA server of the CPU and exchange data.

If you do not select this option, you must add the server interface by entering the "OPC UA communication" entry in the project tree. This interface is then used as OPC UA server interface, see OPC UA server interface configuration (Page 198).

Backward compatible data type definitions according to OPC UA specification ≤ V1.03

The OPC UA specification (<= V1.03) defines mechanisms in order to read out data type definitions, for example for user-defined structures (UDTs), from a server by means of the TypeDictionaries.

In the OPC UA server properties of the CPU, you can set whether the CPU generates these backward compatible data type definitions according to the OPC UA specification ≤ V1.03 for the standard SIMATIC server interface or not.

Because TypeDictionaries are complex and result in large OPC UA XML files (server interfaces) which the client has to interpret, a simpler solution was introduced with OPC UA Specification V1.04 (attribute "DataTypeDefinition" at the data type node). If your client supports the OPC UA specification V1.04 or higher, then disable the option.
9.3.3.3 General settings of the OPC UA server

TCP port for OPC UA

By default, OPC UA uses TCP port 4840. You can, however, select a different port. Entries from 1024 to 49151 are possible. You must, however, make sure that there are no conflicts with other applications. OPC UA clients must use the selected port when establishing a connection.

In the example below, port 48400 is selected:

![Options](image)

Settings for sessions

- Maximum timeout for sessions
  
  In this field, you specify the maximum time period before the OPC UA server closes a session without data exchange.
  
  Possible values between 1 and 600000 seconds.

- Maximum number of OPC UA sessions
  
  In this field, you specify the maximum number of sessions the OPC UA server starts and simultaneously operates.
  
  The maximum number of sessions is dependent on the performance capability of the CPU. Each session ties up resources.

Maximum number of registered nodes

In this field, you specify the maximum number of nodes the OPC UA server registers.

The maximum number of registered nodes depends on the capacity of the CPU and is displayed when you configure the field content (place cursor in field). Each registration ties up resources.

Note

No error message following attempt to register more than the configured maximum number of registrable nodes

If a client tries to register more nodes during runtime than the configured maximum number, the server of the S7-1500 CPU only registers the configured maximum number. From the configured maximum number of registrable nodes, the server sends the client the regular string node IDs so the speed advantage gained by registration for these nodes is lost. The client does not receive an error message.

When configuring, make sure you have a sufficient reserve or have the client calculate the maximum number of nodes allowed before registration.
Additional information

Details on which ports are used by the various services for data transfer via TCP and UDP, and what are the points to note when using routers and firewalls can be found in the FAQ [https://support.industry.siemens.com/cs/ww/en/view/8970169].

9.3.3.4 Settings of the server for subscriptions

Subscription instead of cyclic queries

An alternative to cyclic queries for a PLC tag (polling) is to monitor this value. Use a Subscription: The server informs the client if the value of PLC tags changes. See "The OPC UA client".

One server usually monitors a large number of PLC values. At regular intervals, the server therefore sends the client messages (notifications) containing the new values of the PLC tags.

How frequently does the server send notifications?

When a Subscription is set up, the OPC UA client specifies the intervals at which it wants to be sent the new values in the event of changes. To limit the communication load through OPC UA, set a minimum interval for the messages. For this purpose, use the parameters for the minimum publishing interval and the minimum sampling interval.

Minimum publishing interval

With "Minimum publishing interval", you set the time intervals at which the server sends a message to the client with the new values in the event of changes.

250 ms is used as the "Minimum sampling interval" in the figure below. The value 200 ms is entered as the "Minimum publishing interval".

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum sampling interval:</td>
<td>250 ms</td>
</tr>
<tr>
<td>Minimum publishing interval:</td>
<td>200 ms</td>
</tr>
<tr>
<td>Max. number of monitored items:</td>
<td>2000</td>
</tr>
</tbody>
</table>

In the example, following a value change the OPC UA server will send a new message every 200 ms if the OPC UA client requests an update.

If the OPC UA client requests an update every 1000 ms, the OPC UA server will only send a message with the new values once every 1000 ms (one second).

If the OPC UA client requests an update every 100 ms, the server will still only send a message every 200 ms (minimum publishing interval).
Minimum sampling interval

With "Minimum sampling interval", you set the time intervals at which the OPC UA server records the value of a CPU tag and compares it with the previous value to detect any changes.

If the sampling interval is selected smaller than the publishing interval and an OPC UA client requests a high sampling rate for certain PLC tags, two or more values may be measured during each publishing interval.

In this case, the OPC UA server writes the value changes into the queue and sends all value changes to the client after the completion of the publishing interval. If more value changes occur in the publishing interval than fit in the queue, the OPC UA server overwrites the oldest values. (depending on the set "Discard Policy", the option "Discard Oldest" has to be activated in this case). The most recent values are sent to the client.

Maximum number of monitored elements (monitored items)

In this field, you specify the maximum number of elements that the OPC UA server of the CPU simultaneously monitors for a value change.

The monitoring ties up resources. The maximum number of monitored elements is dependent on the utilized CPU.

Additional information

Information about the system limits of the OPC UA server of the S7-1500 CPUs (firmware V2.0 and V2.1) regarding subscriptions, sampling intervals and publishing intervals can be found in the following FAQ [https://support.industry.siemens.com/cs/ww/en/view/109755846].

When using subscriptions, certain status codes of errors provide information on the error that occurred. For information on causes and remedies for status codes of OPC UA client that appear, see the list of error codes in the online help of Step 7 (TIA Portal) or in the following FAQ [https://support.industry.siemens.com/cs/ww/en/view/109755860].
9.3.3.5 **Handling client and server certificates**

A secure connection between the OPC UA server and an OPC UA client is only established when the server can prove its identity to the client. This is done with the server certificate.

**Certificate of the OPC UA server**

When you have activated the OPC UA server and have confirmed the security prompts, STEP 7 automatically generates the certificate for the server and saves it in the local certificate directory of the CPU. You can view and manage this directory with the local certificate manager of the CPU (exporting or deleting certificates).

The figure below shows the local certificate manager of the CPU with the automatically generated certificate for the OPC UA server:

![Local certificate manager of the CPU](image)

Alternatively, you can also generate a server certificate yourself.

The certificate of the server is transferred from the server to the client during establishment of a connection. The client checks the certificate.
The client user decides whether the server certificate is to be trusted.

The user at the client side now has to decide whether the server certificate is to be trusted. If the user trusts the server certificate, the client stores the server certificate in its directory containing the trusted server certificates.

The following example shows a dialog of the client "UA Sample Client". When the user clicks the "Yes" button, the client trusts the server certificate:

![Figure 9-12 Dialog of the client "UA Sample Client"

See also

Generating server certificates with STEP 7 (Page 183)

Secured transferring of messages

Where does a client certificate come from?

When you use UA clients from manufacturers or the OPC Foundation, a client certificate is generated automatically during installation or upon the first program call. You have to import these certificates via the global certificate manager in STEP 7 and use them for the corresponding CPU (as shown above).

When you program an OPC UA client yourself, you can have the certificates generated by the program; see the section "Instance certificate for the client". Alternatively, you can generate certificates with tools, for example with OpenSSL or the certificate generator of the OPC Foundation:

- The procedure for OpenSSL is described here: "Generating PKI key pairs and certificates yourself".
- Working with the certificate generator of the OPC Foundation is described here: "Creating self-signed certificates".
Announcing client certificates to the server

You need to send client certificates to the server to allow a secure connection to be established.

To do this, follow these steps:

1. Select the "Use global security settings for certificate manager" option in the local certificate manager of the server. This makes the global certificate manager available.

   You will find this option under "Protection & Security > Certificate manager" in the properties of the CPU that is acting as server.

   If the project is not yet protected, select "Security settings > Settings" in the STEP 7 project tree, click the "Protect this project" button and log on.

   The "Global security settings" item is now displayed under "Security settings" in the STEP 7 project tree.

2. Double click "Global security settings".

3. Double click "Certificate manager".

   STEP 7 opens the global certificate manager.

4. Click on the "Trusted certificates" tab.

5. Right-click in the tab on a free area (not on a certificate).

6. Select the "Import" command from the shortcut menu.

   The dialog for importing certificates is displayed.

7. Select the client certificate that the server is to trust.

8. Click "Open" to import the certificate.

   The certificate of the client is now contained in the global certificate manager.

   Note the ID of the client certificate just imported.

9. Click the "General" tab in the properties of the CPU that is acting as server.

10. Click "OPC UA > Server > Security > Secure Channel".

11. Scroll down in the "Secure Channel" dialog to the section "Trusted clients".

12. Double-click in the table on the empty row with "<add new>". A browse button is displayed in the row.

13. Click this button.

14. Select the client certificate that you have imported.

15. Click the button with the green check mark.

16. Compile the project.

17. Load the configuration onto the S7-1500 CPU.

Result

The server now trusts the client. If the server certificate is also considered trusted, the server and client can establish a secure connection.
Accepting client certificates automatically

When you select the option "Automatically accept all client certificates during runtime" (below the "Trusted clients" list), the server automatically accepts all client certificates.

| NOTICE |
| Setting after commissioning |
| In order to avoid security risks, deactivate the "Automatically accept client certificates during runtime" option again after commissioning. |

Configuring security settings of the server

The figure below shows the available server security settings for signing and encrypting messages.

![Figure 9-13 Configuring security settings of the server](image)

Figure 9-13  Configuring security settings of the server
By default, a server certificate is created that uses SHA256 signing. The following security policies are released:

- None
  Insecure end point

**Note**

**Disabling security policies you do not want**

If you have selected all security policies (default setting) in the secure channel settings of the S7-1500 OPC UA server – in other words the end point "No Security" – non-secure data traffic (neither signed nor encrypted) between the server and client is also possible. The identity of the client remains unknown with "No security". Each OPC UA client can then connect to the server irrespective of any subsequent security settings.

When configuring the OPC UA server, make sure that only security policies that are compatible with the security concept for your machine or plant are selected. All other security policies should be disabled.

Recommendation: If possible, use the setting "Basic256Sha256".

- Basic128Rsa15 -Sign
  Insecure end point, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity of the data through signing.

- Basic128Rsa15 -Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.

- Basic256Rsa15 -Sign
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.

- Basic256Rsa15 -Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.

- Basic256Sha256 - Sign
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.

- Basic256Sha256 - Sign & Encrypt
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encryption.
To enable the security setting, click the check box in the relevant line.

**Note**

If you use the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the OPC UA server and OPC UA clients must use "SHA256"-signed certificates.

For the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the certificate authority of STEP 7 automatically signs the certificates with "SHA256".

"No Security" security policy and authentication via user name and password

You can set the following combination:

Security policy = "No Security" and authentication via user name and password.

- The OPC UA server of the S7-1500 supports this combination. OPC UA clients can connect and encrypt the authentication data or not.
- OPC UA client of the S7-1500 CPU also supports this combination: However, in runtime it only connects if it can send the authentication data encrypted via cable!

9.3.3.6 Handling of the client certificates of the S7-1500 CPU

Where does the client certificate come from?

If you are using the OPC UA client of an S7-1500 CPU (OPC UA client enabled), you can create certificates for these clients with STEP 7 V15 and higher as described in the following sections.

When you use UA clients from manufacturers or the OPC Foundation, a client certificate is generated automatically during installation or upon the first program call. You have to import these certificates with the global certificate manager in STEP 7 and use them for the respective CPU.

When you program an OPC UA client yourself, you can have the certificates generated by the program; see the section "Instance certificate for the client". Alternatively, you can generate certificates with tools, for example with OpenSSL or the certificate generator of the OPC Foundation:

- The procedure for OpenSSL is described here: "Generating PKI key pairs and certificates yourself".
- Working with the certificate generator of the OPC Foundation is described here: "Creating self-signed certificates".
Certificate of the OPC UA client of the S7-1500 CPU

A secure connection between the OPC UA server and an OPC UA client is only established if the server classifies the certificate of the client as trusted.

Therefore you have to make the client certificate known to the server.

The following sections describe how you can initially generate a certificate for the OPC UA Client of the S7-1500 CPU and then make it available to the Server.

1. Generate and export a certificate for the client

   For a secure connection you have to generate a client certificate and export the certificate. To do this, follow these steps:

   1. In the "Project tree" area, select the CPU you want to use as a client.
   2. Double-click "device configuration".
   3. In the properties of the CPU click "Protection & Security > Certificate manager".
   4. Double-click "<add>" in the "Device certificates" table.

      STEP 7 opens a dialog.

   5. Click "Add".

   6. Select the "OPC UA client" entry from the list at the "Usage".

      Attention:

      Under "Alternative name of the certificate holder (SAN)" the IP addresses under which the CPU in your system can be accessed has to be entered.

      You must therefore configure the IP interface of the CPU before you have generate a Client certificate.

   7. Click "OK".

      STEP 7 now lists the client certificate in the "Device certificates" table.

   8. Right-click this row and select the "Export certificate" entry from the shortcut menu.

   9. Select a directory in which you store the Client certificate.
2. Announcing the Client certificate to the server

You have to make the Client certificate available to the server to allow a secure connection to be established.

To do this, follow these steps:

1. Select the "Use global security settings for certificate manager" option in the local certificate manager of the server. This makes the global certificate manager available.

   You will find this option under "Protection & Security > Certificate manager" in the properties of the CPU that is acting as server.

   If the project is not yet protected, select "Security settings > Settings" in the STEP 7 project tree, click the "Protect this project" button and log on.

   The "Global security settings" item is now displayed under "Security settings" in the STEP 7 project tree.

2. Double click "Global security settings".

3. Double click "Certificate manager".

   STEP 7 opens the global certificate manager.

4. Click on the "Trusted certificates" tab.

5. Right-click in the tab on a free area (not on a certificate).

6. Select the "Import" command from the shortcut menu.

   The dialog for importing certificates is displayed.

7. Select the client certificate that the server is to trust.

8. Click "Open" to import the certificate.

   The certificate of the client is now contained in the global certificate manager.

   Note the ID of the client certificate just imported.

9. Click the "General" tab in the properties of the CPU that is acting as server.

10. Click "OPC UA > Server > Security > Secure Channel".

11. Scroll down in the "Secure Channel" dialog to the section "Trusted clients".

12. Double-click in the table on the empty row with "<add new>". A browse button is displayed in the row.

13. Click this button.

14. Select the client certificate that you have imported.

15. Click the button with the green check mark.

16. Compile the project.

17. Load the configuration onto the S7-1500 CPU.

Result

The server now trusts the client. If the server certificate is also considered trusted, the server and client can establish a secure connection.
9.3.3.7 Generating server certificates with STEP 7

The description below shows the procedure for generating new certificates with STEP 7 and applies in principle to various uses of the certificates. STEP 7 sets the appropriate purpose - in this case "OPC UA Client & Server" - depending on which area of the CPU properties is used to start the following dialog.

Recommendation: To use the full functionality for the security of the OPC UA server, use the global security settings.

The global security settings are enabled in the CPU properties under "Protection & Security > Certificate manager".

Customizing server certificates

STEP 7 automatically generates a certificate for the OPC UA server of the S7-1500 when you activate the server (see "Activating the OPC UA server (Page 168)". In the process STEP 7 uses the default values for the parameters of the certificate. If you want to change the parameters, follow these steps:

1. Click the Browse button under "General > OPC UA > Server > Security > Secure channel > Server certificate" in the properties of the CPU. A dialog is displayed that shows the certificates available locally.

2. Click the "Add" button.
3. The dialog for generating new certificates is displayed (figure below). The values for an example are already entered:

![Create a new certificate dialog](image)

Figure 9-14 Customizing server certificates

4. Use other parameters if this is necessary in accordance with the security specifications in your company or your customer.
Explanation of fields for certificate generation

- **CA**
  Select whether the certificate is to be self-signed or signed by one of the CA certificates of the TIA Portal. The certificates are described under "Certificates with OPC UA". If you want to generate a certificate that is to be signed by one of the CA certificates of the TIA Portal, the project must be protected and you must be logged in as a user with all the required function rights. Further information can be found under "Basics of user administration in the TIA Portal".

- **Certificate holder**
  The default setting always consists of the name of the project and "OPCUA-1". In the example, the project name is "PLC1". In the properties of the CPU set the project name under "General > Project information" > Name". Keep the default or enter a different name that is more meaningful for the OPC-UA server under "Certificate holder".

- **Signature**
  Here you select the hash and encryption process that is to be used when signing the server certificate. The following entries are available:
  - "sha1RSA",
  - "sha256RSA".

- **Valid from**
  Here you enter the date and time for the beginning of the validity of the server certificate.

- **Valid until**
  Here you enter the date and time for the end of the validity of the server certificate. Ensure that the certificate is valid not only for one year or a few years. In the example the certificate is valid for 30 years. However, for reasons of security you should renew the certificate at much shorter intervals. The long period of validity gives you the opportunity to decide when a suitable moment would be, for example, when the system is being serviced.

- **Usage**
  The default is "OPC UA client & server". Keep this default for the OPC UA server. The "Generate new certificate" dialog can be called from several points in STEP 7. If, for example, you call this dialog for the Web server of the CPU, "Web server" is entered under "Application". The following entries are available in the Usage drop-down list:
  - "OPC UA client"
  - "OPC UA client & server"
  - "OPC UA server"
  - "TLS"
  - "Web server"

- **Alternative name of the certificate holder**
  The following is entered in the example above: "URI:urn:SIMATIC.S7-1500.OPC-UAServer:PLC1,IP:192.168.178.151,IP:192.168.1.1". The following entry would also be valid: "IP: 192.168.178.151, IP: 192.168.1.1". It is important that the IP addresses over which the OPC UA server of the CPU can be accessed are entered here (see "[Access to the OPC UA server](Page 170)"). This allows OPC UA clients to verify whether a connection to the OPC UA server of the S7-1500 is really to be established or whether in fact an attacker is trying to send manipulated values from another PC to the OPC UA client.


9.3.3.8 **Editing the security settings of the OPC UA server.**

OPC UA uses secure connections between client and server. OPC UA checks the identity of the communication partners. OPC UA uses certificates in accordance with X.509-V3 from the ITU (International Telecommunication Union) for client and server authentication.

Exception: A secure connection is not established with the "No security" security policy.

**Configuring security settings of the server**

The figure below shows the available server security settings for signing and encrypting messages.

The security policies are named according to the algorithms used. Example: "Basic256Sha256 - Sign & Encrypt" means: Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.

![Figure 9-15 Configuring security settings of the server](image-url)
By default, a server certificate is created that uses SHA256 signing. The following security policies are released:

- None
  Insecure end point

**Note**

**Disabling security policies you do not want**

If you have selected all security policies (default setting) in the secure channel settings of the S7-1500 OPC UA server – in other words the end point "No Security" – non-secure data traffic (neither signed nor encrypted) between the server and client is also possible. The identity of the client remains unknown with "No security". Each OPC UA client can then connect to the server irrespective of any subsequent security settings.

When configuring the OPC UA server, make sure that only security policies that are compatible with the security concept for your machine or plant are selected. All other security policies should be disabled.

Recommendation: If possible, use the setting "Basic256Sha256".

- Basic128Rsa15 - Sign
  Insecure end point, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity of the data through signing.

- Basic128Rsa15 - Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 128-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.

- Basic256Rsa15 - Sign
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.

- Basic256Rsa15 - Sign & Encrypt
  Secure endpoint, supports a series of algorithms that use the hash algorithm RSA15 and 256-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.

- Basic256Sha256 - Sign
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity of the data through signing.

- Basic256Sha256 - Sign & Encrypt
  Secure endpoint, supports a series of algorithms for 256-bit hashing and 256-bit encryption.
  This endpoint protects the integrity and confidentiality of the data through signing and encrypting.
To enable the security setting click the check box in the relevant line.

**Note**

If you use the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the OPC UA server and OPC UA clients must use "SHA256"-signed certificates.

For the settings "Basic256Sha256 -Sign" and "Basic256Sha256 -Sign & Encrypt", the certificate authority of STEP 7 automatically signs the certificates with "SHA256".

"No Security" security policy and authentication via user name and password

You can set the following combination:

Security policy = "No Security" and authentication via user name and password.

- The OPC UA server of the S7-1500 supports this combination. OPC UA clients can connect and encrypt the authentication data or not.

- OPC UA client of the S7-1500 CPU also supports this combination: However, in runtime it only connects if it can send the authentication data encrypted via cable!
9.3.3.9 User authentication

Types of user authentication

For the OPC UA server of the S7-1500, you can set what authentication is required for a user of the OPC UA client wishing to access the server.

You have the following options:

- **Guest authentication**
  
  The user does not have to prove their authorization (anonymous access). The OPC UA server does not check the authorization of the client user.
  
  If you want to use this type of user authentication, select the "Enable guest authentication" option under "OPC UA > Server > Security > User authentication".

  **Note**
  
  To increase security, you should only allow access to the OPC UA server with user authentication.

- **User name and password authentication**
  
  The user has to prove their authorization (no anonymous access). The OPC UA server checks whether the client user is authorized to access the server. Authorization is given by the user name and the correct password.
  
  If you want to use this type of user authentication, select the "Enable user name and password authentication" option under "OPC UA > Server > Security > User authentication".
  
  Deactivate the guest authentication.
  
  Enter the user in the "User management" table.
  
  To do so, click the "<Add new user>" entry. A new user is created with an automatically assigned name. You can edit the user name and enter the password for the user name. You can add a maximum of 21 users.

- **Additional user administration via the security settings of the project**
  
  If you select this option, the user management for the open project will also be used for user authentication for the OPC UA server: The same user names and passwords are then valid in OPC UA as in the current project.
  
  Proceed as follows to activate user management for the project:
  
  - Click "Security settings > Settings" in the project tree.
  - Click the "Protect this project" button.
  - Enter your user name and your password.
  - Enter additional users under "Security settings > Users and roles".
  
  If you configure an additional OPC UA server in your project, also select the option "Enable additional user administration via the security settings of the project". Repeated input of user names and passwords is then unnecessary.
9.3.3.10 Users and roles with OPC UA function rights

The following options for user authentication use central project settings for project users:

- For the server:
  For configuration of CPU properties (OPC UA > Server > Security > User authentication). Option: "Enable additional user administration via the security settings of the project"

- For the client:
  For configuration of client interface ("Configuration" tab, "Security"). Option: "User (TIA Portal - security settings)"

Requirement

Before you can edit the security settings, the project must be protected and you must be logged on with sufficient rights, for example as administrator.

Settings in the project tree > "Security settings"

You access the central user settings and roles in the protected project in the project tree under "Security settings". This is where you centrally define users with user name, password and function rights. You can simply use these settings elsewhere.

Reusing central security settings

Examples for reusing elsewhere:

- User selection for user authentication for OPC UA server
  With this setting, you tell the server which client (user) with which user name and which password is allowed to access the server.

- User selection for OPC UA client authentication
  With this setting, you tell the client the user name and password that it is to use for client authentication for the server.

The settings for the client and server must correspond: The user name and password used by the client to log on must have been set up on the server and assigned the required authorizations.
Function rights for server and client

The corresponding function rights for the client or the server must also be enabled for users of the client function and users of the server function on an S7-1500 CPU. It is not enough simply to save the user name and password centrally.

Here is an example to illustrate this type of rights use.

1. Under "Security settings > Users and roles", you define a new role in the "Roles" tab with the name "PLC-opcua-role-all-inclusive", for example.
   Tip: The tab may be covered by an information window ("The current status has not yet been checked..."). In this case, first close the information window.

2. In the "Categories of function rights" section, you navigate to the runtime rights and then to the CPU function rights, and select the CPU whose function rights you want to set, for example PLC_2.

3. You will find the following function rights in the "Function rights" section:
   - OPC UA server access
     This function right applies on the OPC UA server of the S7-1500 CPU. Only when this option is selected does a user of the CPU PLC_2 server who has been assigned the role "PLC-opcua-role-all-inclusive" have the following right: For the establishment of a session with the server, the user requires client authentication with one of the user names and corresponding passwords that have been centrally defined (and loaded to the CPU).
   - User authentication of the OPC UA client
     This function right applies on the OPC UA client of the S7-1500 CPU (with client instructions). Only when this option is selected can the user of the client of CPU PLC_2 who has been assigned the role "PLC-opcua-role-all-inclusive" use the user name and password for authentication to establish a session with a server.

4. The role "PLC-opcua-role-all-inclusive" still needs to be assigned to the relevant users ("Users" tab under "Security settings" in the project tree).
Licenses for the OPC UA server

Runtime licenses

A license is required to run the OPC UA server of the S7-1500 CPU. The type of license required depends on the performance of the respective CPU. The following license types are differentiated:

- **SIMATIC OPC UA S7-1500 small** (required for CPU 1511, CPU 1512, CPU 1513, ET 200SP CPUs, CPU 1515SP PC)
- **SIMATIC OPC UA S7-1500 medium** (required for CPU 1515, CPU 1516, Software Controller CPU 1507, CPU 1516pro-2PN)
- **SIMATIC OPC UA S7-1500 large** (required for CPU 1517, CPU 1518)

The required license type is displayed under "Properties > General > Runtime licenses > OPC-UA > Type of required license":

To confirm purchase of the required license, follow these steps:

1. Click "Runtime licenses > OPC UA" in the properties of the CPU.
2. Select the required license from the "Type of purchased license" drop-down list.
9.3.4 Providing methods on the OPC UA server

9.3.4.1 Important facts about server methods

Providing user program for server methods

On the OPC UA server of an S7-1500 CPU (as of firmware V2.5), you have the option of providing methods via your user program. These methods can be used by OPC UA clients, for example to start a manufacturing job using the method call of the S7-1500 CPU.

OPC UA methods, an implementation of "Remote Procedure Calls", provide an efficient mechanism for interactions between different communication nodes. The mechanism provides both job confirmation and feedback values so you no longer have to program handshaking mechanisms.

Using OPC UA methods, you can transfer data consistently without trigger bits/handshaking, for example, or trigger specific actions on the controller.

How does an OPC UA method work?

An OPC UA method in principle operates like a know-how protected function block that is called by an external OPC UA client in runtime.

The OPC UA client only "sees" the defined inputs and outputs. The content of the function block, the method or algorithm, remains hidden to the external OPC UA client. The OPC UA client receives feedback on successful execution and values returned by the function block (method), or an error message if execution has not been successful.

As the programmer, you have full control over and responsibility for the program context in which the OPC UA method runs.

Rules for programming a method and runtime behavior

- Make sure that the values returned by the OPC UA method are consistent with the input values provided by the OPC UA client.

- Follow the rules on assigning name and the structure of parameters, and the permitted data types (see description of the OPC UA server instructions).

- Behavior during runtime: The OPC UA server accepts one call per instance. The method instance is not available for other OPC UA clients until the call has been processed by the user program or has timed out.

The basic procedure for implementing a user program as a server method is set out below.
Implementing a server method
A program (function block) for implementing a server method is structured as follows:

1. Querying the server method call with OPC_UA_ServerMethodPre
   You first call the "OPC_UA_ServerMethodPre" instruction in your user program (i.e. in your server method).
   This instruction has the following tasks:
   - With this instruction you ask the OPC UA server of the CPU whether your server method was called from an OPC UA client.
   - If the method was called and the server method has input parameters, your server method now receives the input parameters.
     The input parameters of the server method come from the calling OPC UA client.

2. Editing the server method
   In this section of the server method, you provide the actual user program.
   You have the same options as in any other user program (for example, access to other function blocks or global data blocks).
   If the server method uses input parameters, these parameters are available to you.
   This section of the server method should only be executed if an OPC UA client has called the server method.
   After successful execution of the method, you set the output parameters of the server method if the method has output parameters.

3. Responding to server method with OPC_UA_ServerMethodPost
   To complete the server method, call the "OPC_UA_ServerMethodPost" instruction.
   Use the parameters to notify the "OPC_UA_ServerMethodPost" instruction whether or not the user program has been processed.
   If the user program has been successfully executed, the OPC UA server is notified via the relevant parameters. The OPC UA server then sends the output parameters of the server method to the OPC UA client.
   Always call the instructions "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost" as a pair irrespective of whether the user program is processed by both instructions or continued in the next cycle.
   You will find an example of a server method implementation in the STEP 7 online help.

Integrating the server method
The diagram below shows how an OPC UA client (A) calls the server method "Cool":
The CPU executes the instance "Cool1" of the server method "Cool" in the cyclic user program ⑥.
The CPU first uses the instruction "OPC_UA_ServerMethodPre" to query ④ whether an OPC UA client has called the server method "Cool" ①.

- If the server method has not been called, program execution returns directly to the cyclic user program over ⑤ and ⑥. The CPU resumes the cyclic user program after "Cool1".
- If the server method has been called, this information is returned to the server method "Cool" over ④. The actual functionality is now executed in the Cool server method, see"<Method Functionality>" in the graphic.
The server method then uses the instruction "OPC_UA_ServerMethodPost" to notify the firmware (B) that the instruction has been executed.

The firmware returns this information over to the calling OPC UA client (A).

The CPU resumes the cyclic user program after "Cool1".

---

**Information about server instructions**

The "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost" are described in detail in the help to the Instructions > Communication > OPC UA > OPC UA server.
9.3.4.2 Boundary conditions for using server methods

Permitted data types

If you provide server methods, observe the following rule:

- Assign the data types as shown below (SIMATIC data type - OPC UA data type). Other assignments are not permitted.

STEP 7 does not check the observance of this rule and does not prevent an incorrect assignment. You are responsible for the rule-compliant selection and assignment of the data types.

You can also use the listed data types, for example, as elements of structures/UDTs for input and output parameters of self-created server methods (UAMethod_InParameters and UAMethod_OutParameters).

<table>
<thead>
<tr>
<th>SIMATIC data type</th>
<th>OPC UA data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>Boolean</td>
</tr>
<tr>
<td>SINT</td>
<td>SByte</td>
</tr>
<tr>
<td>INT</td>
<td>Int16</td>
</tr>
<tr>
<td>DINT</td>
<td>Int32</td>
</tr>
<tr>
<td>LINT</td>
<td>Int64</td>
</tr>
<tr>
<td>USINT</td>
<td>Byte</td>
</tr>
<tr>
<td>UINT</td>
<td>UInt16</td>
</tr>
<tr>
<td>UDINT</td>
<td>UInt32</td>
</tr>
<tr>
<td>ULINT</td>
<td>UInt64</td>
</tr>
<tr>
<td>REAL</td>
<td>Float</td>
</tr>
<tr>
<td>LREAL</td>
<td>Double</td>
</tr>
<tr>
<td>LDT</td>
<td>DateTime</td>
</tr>
<tr>
<td>WSTRING</td>
<td>String</td>
</tr>
<tr>
<td>DINT</td>
<td>Enumeration (Encoding Int32) and all derived data types</td>
</tr>
</tbody>
</table>

User-defined data type required (UDT, user-defined data type)

The user-defined data type must be created with the prefix "Union_", for example "Union_MyDatatype".

The first element (Selector) in this UDT must have the data type "UDINT".
Number of implementable server methods and number of arguments

If you implement server methods via your user program, the number of usable methods is limited depending on the CPU type, see the following table.

<table>
<thead>
<tr>
<th>Technical specification value</th>
<th>CPU 1510SP (F)</th>
<th>CPU 1505 (S/SP/SP F/SP T/SP TF)</th>
<th>CPU 1507S (F)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU 1511 (C/F/T/TF)</td>
<td>CPU 1515 (F/T/TF)</td>
<td>CPU 1517 (F/T/TF)</td>
</tr>
<tr>
<td></td>
<td>CPU 1512C</td>
<td>CPU 1515 SP PC (F/T/TF)</td>
<td>CPU 1518 (F)</td>
</tr>
<tr>
<td>Maximum number of usable</td>
<td>20</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>server methods or max.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of server method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>instances (OPC_UA_ServerMethodPre,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPC_UA_ServerMethodPost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>instructions)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of arguments</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>per method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(More than the specified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of arguments can be</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>configured and loaded into</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the CPU, but an OPC UA client</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cannot call the method)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error message when exceeded

If the maximum number of server methods is exceeded, the OPC_UA_ServerMethodPre or OPC_UA_ServerMethodPost instructions report the error code 0xB080_B000 (TooManyMethods).

Supply of structured data types with nested arrays

If a structured data type (Struct/UDT) contains an array, the OPC UA server does not provide information about the length of this array.

If you use such a structure as the input or output parameter of a server method, for example, you must ensure that the nested array is supplied with the correct length when the method is called.

If you do not adhere to this rule, the method fails with the error code "BadInvalidArgument".
9.3.5 OPC UA server interface configuration

9.3.5.1 What is a server interface?

Definition

A server interface combines related nodes of an OPC UA address space of a CPU into a unit, so that a specific view of this CPU is provided for OPC UA clients.

1st example:

A CPU controls an injection molding machine.

In this example, a server interface contains all OPC UA nodes of the CPU,

- which you can read with an OPC UA client, to get information about this injection molding machine (from readable PLC tags),
- which you can write with an OPC UA client write, to transfer values to the injection molding machine (in writable PLC tags),
- which you can call with an OPC UA client to start injection molding machine functions (via server methods).

This server interface allows a view of a CPU which can be used to control an injection molding machine.

For injection molding machines, the companion specification "Euromap" defines a whole series of OPC UA nodes that you can combine in a server interface.

Other OPC UA nodes of the CPU are not included in this view.

This provides a better overview.

The section "Create server interface" (Page 199) shows how to create a server interface and add related OPC UA nodes to this server interface.

The "Euromap77" serves as an example for related OPC UA nodes.

2nd example

Plant-specific or customer-specific tasks

You can also define a server interface which, for example, meets the requirements in a customer-specific project.

Specify all required OPC UA nodes in an XML file.

Use a tool for this purpose, for example the free program "SiOME" from Siemens; you can find the explanations and download link here [https://support.industry.siemens.com/cs/ww/en/view/109755133].
9.3.5.2 Create a server interface

Create server interfaces

To create an Server interface, follow these steps:

1. Select the project view in the TIA Portal.
2. Select the CPU that you want to use as an OPC UA server.
3. Click "OPC UA communication > Server interfaces".
4. Double-click "Add new server interface".

   STEP 7 creates a new server interface, names it "Server interface_1" and opens the
   editor for server interfaces.
5. Change the name of the new server interface so that it is descriptive in your project.
6. Click the "Import companion specification" button in the top right of the Server Interfaces
   view.

   STEP 7 displays the "Import" dialog.
7. Select an XML file which contains an instance (or multiple instances) of a companion
   specification.

   The "Using OPC UA companion specification" section describes how to
   create such an XML file.

   STEP 7 imports the XML file and displays the new server interface in the editor.

   The following figure shows a section of a server interface which uses an instance of the
   "Euromap" companion specification:
**Information on the server interface**

The editor for server interfaces is structured as a table and provides a range of information about the imported node set file (OPC UA-XML file).

The tables provide the following information:

- **Name of the node**
  The top node is named "IMM_manufacturer_01234" in the example. This name stands for the injection molding machine as a whole. It is the name of the instance of the "Euromap" companion specification that is used here.

  According to the companion specification, the instance name should begin with "IMM", followed by the name of the manufacturer of the injection molding machine; the serial number of the machine is added to the end. This allows a unique identification of the machine.

  The name of this node is assigned by the user.

  The names of all other (lower-level) nodes are defined by the specification (in the example above by the "Euromap" companion specification). These node names must not be changed by the user. This ensures a uniform view of all injection molding machines, which complies with the specification.

- **Node type**
  Type of the node. The type is specified by the companion specification that is used.

  STEP 7 shows a node type in red in the table when the imported XML file contains no type definition.

  In this case, you can find the referenced but missing namespaces in the properties of the server interface under "Namespaces".

  Import the referenced XML files that contain the missing namespaces (and all data types defined in them) into a server interface to be created.

- **Access level**
  Nodes (tags) can only be readable (RD) or readable and writeable (RD/WR).

  The server methods of a specification can always be called. If they were not callable, they would not have been included in the specification.

- **Description**
  The imported XML file may contain descriptions of the nodes in several languages.

  If you have imported such a file, STEP 7 displays the description in the project language.

  If no description is available in the XML file in the project language, then the English description is displayed.
• **Local data**

STEP 7 displays the data block which is assigned to the node in the CPU (mapping). The CPU reads the value of the OPC UA node from this data block. The OPC UA server then makes this value available for OPC UA clients.

If a data block is displayed in red, then the specified data block is not available in the CPU.

In this case you have to create the missing data block in the CPU (of user program) and supply it with a value.

Alternative: Change the mapping of the OPC UA node to a local data block (or to a tag of a tag table of your user program) in the XML file.

The "Using OPC UA companion specification (Page 202)" section describes how to change mapping.

• **Data type**

This is the data type of the data block in the CPU, from which the value of an OPC UA node is read, or to which a value is written.

**Consistency check**

You have the option of checking the server interface. STEP 7 hereby checks whether the nodes (e.g. tags) are mapped correctly in the imported node set file (OPC UA XML file).

1. Select the nodes that you want to check.

2. Click "Consistency check".

The consistency check checks the mapping of the local data against the displayed server interface:

- Do the names of the PLC tags in the CPU program correspond to those of the imported server interface?

- Do the data types of the PLC tags correspond to the corresponding nodes in the server interface? STEP 7 displays corresponding warnings and errors if the above conditions are not met.
Using OPC UA companion specifications

Introduction

OPC UA is universally applicable: The standard itself does not, for example, specify how PLC tags are to be named. It is also up to the individual user (application developer) to program and name server methods that can be called over OPC UA.

Information modeling and standardization for devices and sectors

For applications of the same kind, it is worth standardizing your device or machine interface with the "OPC UA toolkit".

Many different bodies and working groups have driven forward standardization and developed a range of companion specifications.

These specifications define:

- The objects, methods and tags with which a typical device or machine is to be described.
- The namespace intended for the specified objects.

Machines are typically structured in functional or technological units, and these units are then standardized.

Companion specifications offer machine and plant operators the benefits of a standardized interface. For example, all RFID readers that comply with the AutoID specifications can be integrated in the same way. This means that all RFID readers that comply with the AutoID specifications can be addressed by OPC UA clients in the same way irrespective of manufacturer.

Another example of companion specifications is the Euromap 77 Companion Specifications from the injection molding machinery sector.

The following section uses the example of Euromap 77 to detail how to apply companion specifications in STEP 7 (TIA Portal) and create the necessary PLC tags.

Example: Euromap 77

Euromap 77 standardizes the exchange of data between injection molding machines and the higher-level MES (manufacturing execution system). This allows the MES to connect all lower-level injection molding machines in the same way.

The standardized data interface facilitates the integration of injection molding machines into a plant.
Using companion specifications: Overview

Euromap 77 is described in the OPC UA XML file "Opc_Ua.EUROMAP77.NodeSet2.xml".

Note

Euromap 77, Euromap 83 and OPC UA for Devices (DI)

With Release Candidate 2, some of the Euromap definitions have been transferred from Euromap 77 to Euromap 83. You will therefore also need to import the OPC UA server interface of Euromap 83.

"OPC UA for Devices" is a generally applicable information model for the configuration of hardware and software components. The information model also serves as the basis for other companion standards and is therefore also imported.

The OPC UA XML files are available here:

Euromap77 [http://www.euromap.org/euromap77]
Euromap83 [http://www.euromap.org/euromap83]
OPC UA for Devices [https://opcfoundation.org/UA/schemas/DI/]

The Euromap OPC UA XML files define the interface of the OPC UA server of an injection molding machine that complies with the Euromap 77 companion standard.

Proceed as follows to model the OPC UA server of the S7-1500 CPU in accordance with the Euromap 77 standard:

1. Generate an XML file by creating an instance of the type "IMM_MES_InterfaceType".
   "IMM_MES_InterfaceType" is the highest-level node in Euromap 77: This data type is directly derived from the OPC UA data type BaseObjectType".
   All Euromap 77 (83) tags and methods are defined under "IMM_MES_InterfaceType". "Step 1" sets out the procedure in detail.

2. Assign PLC tags and FB instances (server methods) from your S7-1500 CPU to the tags and methods of Euromap 77 (the Euromap 77 information model).
   "Step 2" sets out the procedure in detail.

3. Import the XML file as "Server interface".
   "Step 3" sets out the procedure in detail.
   Compile the STEP 7 project.
   Download the project to the CPU that acts as the controller for an injection molding machine.

4. In your STEP 7 project, create the PLC tags and server methods to which you have assigned Euromap 77 tags and methods under 2.
   The PLC tags must have compatible data types; see "Mapping data types".
   Enable read and write access to these PLC tags for OPC UA clients in accordance with Euromap 77.
   Save the PLC tags in a data block, for example. "Step 4" sets out the procedure in detail.

Result: The Euromap 77 tags and server methods are available for OPC UA clients in the address space of the OPC UA server of your CPU.
Step 1: Creating an instance

The following section describes how to use the free program "SiOME", the "Siemens OPC UA Modelling Editor".

With SiOME, you can create an OPC UA XML file, which describes the server interface (an information model).

To load and publish the new interface into the OPC UA server of an S7-1500 CPU, import the server interface into the STEP 7 project, see section "Create server interface (Page 199)".

When the project is loaded into the CPU, the new server interface is available for OPC UA clients.

Other tools for designing information models

You can also work with other tools to design information models to create a server interface, such as the program "UaModeler" from Unified Automation. You can download the "UaModeler" program from here [https://www.unified-automation.com/downloads/opc-ua-development.html].

The tools for designing information models are constantly being honed and improved. You should therefore always use the documentation provided by the manufacturer.

SiOME

With the help of SiOME (Siemens OPC UA Modeling Editor), a tool for implementing OPC UA companion specifications, you can design information models / address spaces for your OPC UA server and can create new types and instances of OPC UA nodes.

You can also use SiOME to map UA tags to PLC tags and UA methods to PLC function blocks (instances).

Download link and explanations about SIOME are available here [https://support.industry.siemens.com/cs/ww/en/view/109755133].

Proceed as follows to create an XML file with an instance of "IMM_MES_InterfaceType":

1. Download the files "Opc_Ua.EUROMAP77.NodeSet2.xml" and "Opc_Ua.EUROMAP83.NodeSet2.xml" from the Euromap website (see above).
   Euromap77/83 is an OPC UA Companion specification, this means that it is based on OPC UA and uses data types defined in the file "Opc.Ua.Di.NodeSet2.xml".
3. Start SiOME.
4. First, import the namespace "http://opcfoundation.org/UA/DI/".
   To do so, click the "Import XML" button in the "Information model" area.

   Select the file "OPCUaDiNodeSet2.xml" and click "Open" to import the file.

   SiOME imports the XML file and shows the namespace "http://opcfoundation.org/UA/DI/
   in the "Namespaces" area.

   The standard namespace "http://opfoundation.org/UA/" is always available in SiOME and does not have to be imported.

5. Now import the namespace "http://www.euromap.org/euromap83/"
   To do so, click the "Import XML" button in the "Information model" area.

   Select the file "Opc_Ua.EUROMAP83.NodeSet2.xml".

   SiOME imports the XML file and shows the namespace "http://www.euromap.org/euromap83/
   in the "Namespaces" area.

6. Now import the namespace "http://www.euromap.org/euromap77/"
   To do so, click the "Import XML" button again in the "Information model" area.

   Select the file "Opc_Ua.EUROMAP77.NodeSet2.xml".

7. **Defining your own namespace and creating an instance**
   So far, you have imported Euromap77.

   Now use the companion specification for an injection molding machine, i.e. you create an instance of Euromap77 (a usage).

   In the "Namespaces" area, right-click "Models" and select "Add Model".

   SiOME opens the "Add Model" dialog.

8. Click "New Model", to create a new model.

    SiOME now shows in green the switch in front "New Model":

    ![Add Model](image)
9. Add your own namespace to the XML file.
   Use a unique name for this purpose.
   In the example, enter "YourCompany.com":

   **Namespaces**

   - OPC UA Modelling Editor Project
     - Models
       - 0: http://opcfoundation.org/UA/
       - 1: http://opcfoundation.org/UA/01/
       - 2: http://www.euromap.org/euromap83/
       - 3: http://www.euromap.org/euromap77/
       - 4: YourCompany.com

10. Create new sequence
    Now create now a new instance of Euromap77. This allows you to use the Euromap77 for an injection molding machine.
    Right-click on "Objects" in the "Information model" area.
    SIOME shows a shortcut menu.
11. Click "Add Instance" to add a new instance.
    SIOME displays the "Add Instance" dialog.
    At "Name" insert a descriptive name for your instance, i.e. for the machine you are using for Euromap77.
    In the example, enter "IMM_Manufacturer_01234".
    For "TypeDefinition", select "IMM_MES_InterfaceType".
    Click "OK".
    SIOME shows the new instance in the "Information model" area under "Objects":

   **Information model**
12. Click the black arrow in front of "IMM_Manufacturer_01234", to open the instance:

13. Now create a new instance for "InjectionUnits". Right-click "InjectionUnits". SiOME displays the "Add Instance" dialog. At "Name", insert a descriptive name for your instance. In the example, enter "InjectionUnit_1". For "TypeDefinition", select "InjectionUnitType". Click "OK".

14. Now create a new instance each for "Molds" and "PowerUnits", as described above for "InjectionUnits".

15. Now save the XML file. Export the file for this purpose. Click the "Export XML" button in the "Information model" area.

16. Leave the "Include mappings" option unselected, because the file still contains no mappings (UAtags on PLC tags, UAmethods on server methods of the CPU). The imported namespaces are required for Euromap77. Therefore, SiOME displays these in grey. Click "OK".

17. Select a descriptive name for the XML file and save the file.

Result:
You have now created an XML file containing the companion specification "Euromap77" and the instances you added. Euromap77 is used for this purpose.
Step 2: Assigning PLC tags

The description below uses the example of a tag to show how you assign specific PLC tags and server methods to the tags and methods of Euromap 77.

The easy way

We recommend using the SiOME tool here as well. You have to enable mapping in SiOME for this purpose:

In the "TIA Portal" area, select the data blocks whose tags you want to assign ("map") to the OPC UA tags of the information model.

The tags from the TIA Portal project can easily be connected to the OPC UA tags using drag-and-drop.

To link methods, you must use drag-and-drop to link the instance data blocks of the corresponding TIA Portal function blocks to the methods of the information model.

When exporting, select the "Include mappings" option, since the information model for an OPC UA server is required here.

This option is not required if you need the information model for an OPC UA client.

A detailed description can be found in the online support application example (OPC UA for Devices [https://opcfoundation.org/UA/schemas/DI/]).
Assigning PLC tags manually

If you do not assign the data with SiOME, proceed as follows:

1. Use an editor to open the file "demo.xml" that you created in "Step 1".
   In the XML file, search for the UA tag with
   NodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits.NodeVersion".
   These tags are assigned a PLC tag in the next step.

2. To assign this OPC UA tag a PLC tag, add an extension to the XML element of the OPC
   UA tag.
   In turn, you then add an element of the type "<si:VariableMapping>" to this extension:

   ```xml
   <UANodeSet xmlns="http://www.siemens.com/OPCUA/2017/SimaticNodeSetExtensions">
     <NodeSet>
       <VariableMapping Name="MyIMM_MES_Interface.InjectionUnits.NodeVersion">
         <DisplayValue>DataType="String" ParentNodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits" NodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits.NodeVersion" BrowseName="NodeVersion">
           <DisplayName>NodeVersion</DisplayValue>
           <Reference ReferenceType="HasProperty" IsForward="false">ns=1;s=MyIMM_MES_Interface.InjectionUnits</Reference>
           <Reference ReferenceType="HasTypeDefinition">1-68</Reference>
           <Extensions>
             <Extension>
               <si:VariableMapping>MyIMM_MES_Interface."InjectionUnits.NodeVersion"
             </si:VariableMapping>
           </Extension>
         </VariableMapping>
       </NodeSet>
     </UANodeSet>
   </UANodeSet>
   ```

   3. The XML element "<si:VariableMapping>" is defined in the XML namespace
      "http://www.siemens.com/OPCUA/2017/SimaticNodeSetExtensions".
      You therefore need to assign this namespace to the XML element "<UANodeSet>",
      for example in a one-off operation with the following code line at the start of the XML file:
      ```xml
      ```
      If you do not add the namespace, the element "<si:VariableMapping>" will not be known
      in the file.

4. Save the "demo.xml" file and close the editor.
Useful information: Assigning PLC methods

As well as assigning tags, you can also assign methods to an FB instance (user program or function block as representation of the method).

To assign an OPC UA method to an FB instance, you also need to add an extension to the OPC UA XML file following the procedure outlined below. Please note that the suffix ".Method" without quotation marks must be added to the instance name.

You add an element of the <si:MethodMapping> type to the extension:

The properties of an OPC UA method with the BrowseName "InputArguments" and "OutputArguments" are OPC UA tags and are not assigned.

```
<UAMethod ParentNodeId="ns=1;i=5003" NodeId="ns=1;i=MyIM_S_Interface_MachineConfiguration.SetServerTime" BrowseName="4:SetServerTime"
MethodDeclarationId="ns=0;i=2005">
  <DisplayName>SetServerTime</DisplayName>
  <References>
    <Reference ReferenceType="HasProperty">
      ns=1;s=MyIM_S_Interface_MachineConfiguration.SetServerTime.InputArguments</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5003</Reference>
  </References>
  <Extensions>
    <Extension>
    </Extension>
  </Extensions>
</UAMethod>
```

Step 3: Importing server interfaces

Proceed as follows to import an OPC UA XML file as "Server interface":

1. Open the STEP 7 project.
2. Click "OPC UA communication > Server interfaces".
3. Double-click on "Import server interface".
4. In the "Import" dialog, select the file that you want to import as server interface.
   In the example, this is the file "demo.xml".
5. Click the "Import" button.
   The "demo.xml" file relates to the files just specified.
   Always use the latest versions of these files as STEP 7 does not compile OPC UA XML files with errors.

The above image of the machine generated in accordance with Euromap 77 specifications is now available in the address space of the OPC UA server.
Step 4: Creating PLC tags and server methods in the STEP 7 project

In the example, you have assigned the PLC tag "MyIMM_MES_Interface.InjectionUnits.NodeVersion" to the UA tag with NodeId="ns=1;s=MyIMM_MES_Interface.InjectionUnits.NodeVersion"; see "Step 2: Assigning PLC tags".

Proceed as follows to create the required PLC tag with STEP 7 (TIA Portal):

1. Generate a "MyIMM_MES_Interface" data block.
2. Generate the DB element "InjectionUnits.NodeVersion". Use the SIMATIC data type compatible with the OPC UA data type. The UA tag in the example has the OPC UA data type "String" (DataType="String"). The compatible SIMATIC data type is WSTRING.
3. Compile the project.
4. Download the project to the CPU.

Importing exported OPC UA XML files to an S7-1500 CPU

Please note the following information when importing server interfaces that come from the OPC UA XML export of an S7-1500.

Note

Import blocked for namespace "http://www.siemens.com/simatic-s7-opcua"

You cannot import server interfaces with the namespace "http://www.siemens.com/simatic-s7-opcua" to an S7-1500 CPU because this namespace is reserved for S7-1500 CPUs (standard SIMATIC server interface) and is not available for imports.

If you want to import a server interface with the namespace "http://www.siemens.com/simatic-s7-opcua", open the server interface to be imported (OPC UA XML file) and change the namespace in the relevant places. The file thus changed can then be imported.
### Integrity of the OPC UA XML files

OPC UA XML files represent the server address space. These files are, for example, imported by you in the context of OPC UA Companion specifications as a server interface after adaptation to the application, loaded with the hardware configuration into the S7-1500 CPU and tested.

<table>
<thead>
<tr>
<th>WARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No checking of imported OPC UA XML files</strong></td>
</tr>
<tr>
<td>Protect these OPC UA XML files against unauthorized manipulation since STEP 7 does not check the integrity of these files.</td>
</tr>
</tbody>
</table>

**Recommendation**

To minimize risks in the case of an extension or adaptation of the server address space, follow these steps:

1. Protect the project (project navigation: Security settings > Settings).
2. Export the corresponding server interface before the extension or adaptation.
3. Revise this OPC UA XML file.
4. Import the file again as a server interface.

### Mapping data types

The table below shows the compatible SIMATIC data type for each OPC UA data type.

Assign the data types as shown below (SIMATIC data type – OPC UA data type). Other assignments are not permitted. STEP 7 does not check the observance of this rule and does not prevent an incorrect assignment. You are responsible for the rule-compliant selection and assignment of the data types.

You can also use the listed data types, for example, as elements of structures/UDTs for input and output parameters of self-created server methods (UAMethod_InParameters and UAMethod_OutParameters).

**Table 9-2 Mapping of data types**

<table>
<thead>
<tr>
<th>SIMATIC data type</th>
<th>OPC UA data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>Boolean</td>
</tr>
<tr>
<td>SINT</td>
<td>SByte</td>
</tr>
<tr>
<td>INT</td>
<td>Int16</td>
</tr>
<tr>
<td>DINT</td>
<td>Int32</td>
</tr>
<tr>
<td>LINT</td>
<td>Int64</td>
</tr>
<tr>
<td>USINT</td>
<td>Byte</td>
</tr>
<tr>
<td>UINT</td>
<td>UInt16</td>
</tr>
<tr>
<td>UDINT</td>
<td>UInt32</td>
</tr>
<tr>
<td>ULINT</td>
<td>UInt64</td>
</tr>
</tbody>
</table>
### User-defined data type for UNION required

The figure below shows the tag "MyVariable", which has the data type "Union_MyDatatype". This SIMATIC data type corresponds to an OPC UA tag with the data type UNION.

The figure shows an example of the declaration: When Selector = 1, Union accepts one ByteString; when Selector = 2, Union accepts one WString.

<table>
<thead>
<tr>
<th>SIMATIC data type</th>
<th>OPC UA data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL</td>
<td>Float</td>
</tr>
<tr>
<td>LREAL</td>
<td>Double</td>
</tr>
<tr>
<td>LDT</td>
<td>DateTime</td>
</tr>
<tr>
<td>WSTRING</td>
<td>String</td>
</tr>
<tr>
<td>DINT</td>
<td>Enumeration (Encoding Int32) and all derived data types</td>
</tr>
</tbody>
</table>

User-defined data type required (UDT, user-defined data type)

The user-defined data type must be created with the prefix "Union_", for example "Union_MyDatatype"; see example below the table.

The first element (Selector) in this UDT must have the data type "UDINT".

The figure below shows the tag "MyVariable", which has the data type "Union_MyDatatype". This SIMATIC data type corresponds to an OPC UA tag with the data type UNION.

The figure shows an example of the declaration: When Selector = 1, Union accepts one ByteString; when Selector = 2, Union accepts one WString.
Using other OPC UA basic data types

Apart from the OPC UA data types listed in the section "Mapping of data types" and their correspondences on the SIMATIC side, there are the following OPC UA basic data types which you can also use:

- OpcUa_Nodeld
- OpcUa_QualifiedName
- OpcUa_Guid
- OpcUa_LocalizedText
- OpcUa_ByteString
- OpcUa_XmlElement

Requirement for the use of the basic data types listed above as variables in the application program: The basic data types have to exist as complex data types that are structured exactly like the corresponding OPC UA basic data types.

- OpcUa_Nodeld and OpcUa_QualifiedName exist as system data types; that's why you can use these data types not only for single variables but also as elements of a structure.
- For the remaining basic data types, you have to create a PLC data type in accordance with the OPC UA specification and subsequently use it as an element in a structure so that the data types of the elements can be resolved. What each PLC data type must look like is described below for every single basic data type. "EUInformation" is an example of a data structure in which, for example, the UDT "LocalizedText" is used. EUInformation contains information on EngineeringUnits. You can find an example of the implementation of the EUInformation data structure at the end of the PLC data type descriptions.

System data type "OPC_UA_Nodeld"

For the OPC UA basic data type "OpcUa_Nodeld", please refer to the following table for the meaning of the parameters. Use OPC_UA_Nodeld for the identification of a node in the OPC UA server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>S7 data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NamespaceIndex</td>
<td>UINT</td>
<td>Namespace index of the node in the OPC UA server. A node can, for example, be a tag.</td>
</tr>
<tr>
<td>Identifier</td>
<td>WSTRING[254]</td>
<td>The designation of the node (object or tag) depends on the identifier type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Numeric identifier: The node is labeled with a number, for example &quot;12345678&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• String identifier: The node is labeled with a name, for example &quot;MyTag&quot;. No distinction is made between upper and lower case.</td>
</tr>
<tr>
<td>IdentifierType</td>
<td>UDINT</td>
<td>Type of identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Numeric identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: String identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: GUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: Opaque</td>
</tr>
</tbody>
</table>
System data type "OPC_UA_QualifiedName"

See the following table for the structure of the system data type "OPC_UA_QualifiedName":

<table>
<thead>
<tr>
<th>Name</th>
<th>S7 data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NamespaceIndex</td>
<td>UINT</td>
<td>The namespace index of the name.</td>
</tr>
<tr>
<td>Name</td>
<td>WSTRING[64]</td>
<td>Name of the node or tag.</td>
</tr>
</tbody>
</table>

UDT "Guid"

For the basic data type "Guid", create the following PLC data type. The default values used as examples can also be set differently.

<table>
<thead>
<tr>
<th>Guid</th>
<th>Data type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Date 1</td>
<td>UDInt 16#11223344</td>
</tr>
<tr>
<td>2</td>
<td>Date 2</td>
<td>Uint 16#5566</td>
</tr>
<tr>
<td>3</td>
<td>Date 3</td>
<td>Uint 16#7788</td>
</tr>
<tr>
<td>4</td>
<td>Date 4</td>
<td>ULIInt 16#99ABCCDDEEFF11</td>
</tr>
</tbody>
</table>

UDT "LocalizedText"

For the basic data type "LocalizedText", create the following PLC data type:

<table>
<thead>
<tr>
<th>LocalizedText</th>
<th>Data type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EncodingByte</td>
<td>Byte</td>
<td>16#3</td>
</tr>
<tr>
<td>Locale</td>
<td>WString[5]</td>
<td>WSTRING&quot;de-DE&quot;</td>
</tr>
<tr>
<td>Text</td>
<td>WString[255]</td>
<td>WSTRING&quot;Text&quot;</td>
</tr>
</tbody>
</table>

The EncodingByte indicates which fields (Locale or Text) are available:

<table>
<thead>
<tr>
<th>EncodingByte</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The fields Locale and Text are empty</td>
</tr>
<tr>
<td>1</td>
<td>The field Locale has content, the field Text is empty</td>
</tr>
<tr>
<td>2</td>
<td>The field Locale is empty, the field Text has content</td>
</tr>
<tr>
<td>3</td>
<td>The fields Locale and Text have content</td>
</tr>
</tbody>
</table>
UDT "ByteString"

For the basic data type "ByteString", create the following PLC data type; in this case, for example, a ByteString array with 12 elements:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteString</td>
<td>Array[0..11] of Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[0]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[1]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[2]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[3]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[4]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[5]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[6]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[7]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[8]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[9]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[10]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
<tr>
<td>ByteString[11]</td>
<td>Byte</td>
<td>16#0</td>
</tr>
</tbody>
</table>

UDT "XmlElement"

An XmlElement is a serialized XML fragment (UTF-8 string)

For the basic data type "XmlElement", create the following PLC data type:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmlElement</td>
<td>WString</td>
<td>&quot; WSTRING#1&quot;</td>
</tr>
</tbody>
</table>

Example: Structure of EUInformation with UDT "LocalizedText"
MinimumSamplingInterval attribute of tags

In addition to "Value", "DataType" and "AccessLevel", you can also set the "MinimumSamplingInterval" attribute for a tag in the XML file that represents the server address space.

The attribute specifies how fast the server can sample the tag value.

The OPC UA server of the S7-1500 CPU handles the values for MinimumSamplingInterval as follows:

- Negative values and values greater than 4294967 are set to -1 (meaning: Default setting of the server should be used for the sampling interval for sampling. This is defined by the publishing interval of the subscription, see OPC UA specification part 4).
- Decimal numbers are rounded to three decimal places.

9.3.5.4 Missing namespaces

Introduction

A node set file (OPC UA XML file) may contain references to other OPC UA XML files that provide type definitions in other namespaces. You must import all required definitions for the information model in order for the server interface to work properly.

Example for list of namespaces (NamespaceUris) in an OPC UA XML file

```xml
<NamespaceUris>
    <Uri>http://www.siemens.com/SimaticIdent/RF600R/</Uri>
    <Uri>http://opcfoundation.org/UA/DI/</Uri>
    <Uri>http://opcfoundation.org/UA/AutoID/</Uri>
</NamespaceUris>
```

Importing additional namespaces

If namespaces are still missing after the import of an OPC UA XML file, STEP 7 generates an error message.

In this case, import the OPC UA XML files with the missing namespaces into a new server interface. You must create a new server interface, otherwise the additional imported XML file would overwrite the current companion specification (the current information model).

The missing namespaces are displayed in red in the properties of the server interface.

After importing the XML files, you must click the "Refresh interface" button to refresh the view.
9.3.5.5 Coordinating write and read rights for CPU tags

Definition of write and read rights in the information model (OPC UA XML)

In the OPC UA information model, the attribute "AccessLevel" regulates access to tags. AccessLevel is defined bit by bit:

Bit 0 = CurrentRead and Bit 1 = CurrentWrite. The meaning of the bit combinations is as follows:

- AccessLevel = 0: no access
- AccessLevel = 1: read only
- AccessLevel = 2: write only
- AccessLevel = 3: read+write

Example of the assignment of write and read rights (read+write)

```xml
<UAVariable NodeId="ns=3;s="&quot;Data_block_2&quot;.&quot;Static_1&quot;"
BrowseName="3:Static_1"
ParentNodeId="ns=3;s="&quot;Data_block_2&quot;"
DataType="INT"
AccessLevel="3">
<DisplayName>Static_1</DisplayName>
</UAVariable>
```

Definition of write and read rights in STEP 7

When you define tags, you specify the access rights using the properties "Accessible from HMI/OPC UA" and "Writable from HMI/OPC UA".

Example of the assignment of write and read rights

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Accessible from HMI/OPC UA</th>
<th>Writable from HMI/OPC UA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static_1</td>
<td>Int</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Interaction between write and read rights

If you have imported an OPC UA server interface and AccessLevel attributes are set in this OPC UA XML file, the write and read rights are defined by the following rule: The least extensive access rights for each setting apply.

Example

- AccessLevel = 1 (read only) in the OPC UA server interface
- Both "Accessible from HMI/OPC UA" and "Writable from HMI/OPC UA" are selected in the PLC tag table

Result: The tag is only read.
**Rules**

If write rights are required:
- AccessLevel = 2 oder 3
- "Writable from HMI/OPC UA" enabled

If read rights are required:
- AccessLevel = 1 (AccessLevel 3 is also possible, but misleading. The settings suggests that an OPC UA client has write and read rights)
- "Accessible from HMI/OPC UA" enabled, "Writable from HMI/OPC UA" disabled

If neither read nor write rights are to be granted (no access):
- AccessLevel = 0
- "Accessible from HMI/OPC UA" disabled

Only one of the two conditions needs to be met to block all access. In this case, review whether the tag in the OPC UA server interface is actually necessary at all.

**Access table**

"Accessible from HMI/OPC UA" must be set if access over OPC UA is to be possible at all. "Writable from HMI/OPC UA" must be set to allow an OPC UA client to write a tag / DB element.

Please see the table for the resulting access right.

<table>
<thead>
<tr>
<th>OPC UA XML</th>
<th>STEP 7 (TIA Portal), for example tag table</th>
<th>Resulting access right</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessLevel</td>
<td>Accessible from HMI/OPC UA</td>
<td>Writable from HMI/OPC UA</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>Enabled</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>3</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>2</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>3</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

(x = don't care)
9.3.5.6 Consistency of CPU tags

"AccessLevelEx" attribute extends access properties

As of firmware version V2.6, the OPC UA server of the S7-1500 CPU supports not only the attribute "AccessLevel" (see Coordinating write and read rights for CPU tags (Page 218)) but also the attribute "AccessLevelEx" which, in addition to the already explained bits for read access and write access, provides information on the consistency of a OPC UA tag. The new attribute was introduced with version V1.04 of the OPC UA specification (Part 3, Address Space Model).

Reading consistency properties

In the OPC UA information model of the OPC UA server, the attribute "AccessLevel" defines the access.

AccessLevelEx is defined bit by bit; in this case, the relevant bits are:

- Bit 0 = CurrentRead
- Bit 1 = CurrentWrite
- Bits 2 to 7 are not relevant for the OPC UA server of an S7-1500 CPU

The meaning of the bit combinations is explained in the section on read and write rights.

The following bits for consistency are also added:

- Bit 8 = NonatomicRead; the bit is set if the tag cannot be read consistently. For read consistency of tags, bit 8=0.
- Bit 9 = NonatomicWrite; is set if the tag cannot be written consistently. For write consistency of tags, or if no write access is granted, bit 9=0.

Examples

An OPC UA tag (structure) is readable and writable; but inconsistent for reading and writing access.

Consequently: Bits 0, 1, 8 and 9 are set: AccessLevelEx = "771" (1+2+256+512).

Another structure is read-only.

Consequently: Bits 0 and 8 are set, bit 1 and bit 9 are not set: AccessLevelEx = "257" (1+0+256+0).
Handling of the attribute in the server

The "AccessLevelEx" attribute is only available in the OPC UA server. The attribute is not present in a node set file (XML export file).

However, the attribute "AccessLevel", which is exported, includes the information from "AccessLevelEx", see next section.

Export

With XML export of the standard Simatic server interface, the server sets the attribute "AccessLevel", which was extended to 32 Bit Advanced in V1.04 compared to V1.03, to the value of attribute "AccessLevelEx".

Import

When importing a node set file (e.g. from an export of a server interface), the S7-1500 CPU sets the attribute "AccessLevelEx" according to its own estimate of the consistency of the imported data type, see next section. The imported value is ignored.

Consistency of data types at the server interface

The consistency of tags ("atomicity" in the language usage of OPC UA) within a program cycle of an S7-1500 CPU is ensured at the nodes of the server interface for the following data types:

- BOOL, BYTE, WORD, DWORD, LWORD
- SINT, INT, LINT, DINT, USINT, UINT, ULINT, UDINT
- REAL, LREAL
- DATE, LDT, TIME, LTIME, TIME_OF_DAY, LTIME_OF_DAY, S5TIME
- CHAR, WCHAR
- System data types and hardware data types that are based on the above-mentioned data types are also consistent.
  
  Example: HW_ANY, derived from UINT (U1nt16).

**Tip:** If you browse in the address space of the S7-1500 CPU (e.g. with the OPC UA Client UaExpert), you can find the consistent data types under Types > BaseDataType > Enumeration/Number/String.

Tags of the following data types are **not** consistent ("nonatomic" in the language usage of OPC UA):

- SIMATIC structures are generally not consistent. This means that all tags which, for example, have unknown structures or a UDT data type are not consistent.
- System data types such as DTL, IEC_Counter, IEC_TIMER, etc. are data types that are derived from structures.

**Tip:** If you browse in the address space of the S7-1500 CPU (e.g. with the OPC UA Client UaExpert), you can find data types based on structures under Types > BaseDataType > Structure.
9.3.5.7 Notes on configuration limits when using server interfaces

When you use OPC UA server interfaces, you must comply with limits for the following objects in line with the S7-1500 CPU performance class:

- Number of server interfaces
- Number of OPC UA nodes
- Load object data volume
- If you have implemented methods: Number of server methods or server method instances

Configuration limits for OPC UA server interfaces and methods

The table below sets out the configuration limits for S7-1500 CPUs; these must also be taken into account when you compile and load a configuration.

A violation of configuration limits results in an error message.

<table>
<thead>
<tr>
<th>Technical specification value</th>
<th>CPU 1510SP (F)</th>
<th>CPU 1511 (C/F/T/TF)</th>
<th>CPU 1505 (S/SP/SP F/SP T/SP TF)</th>
<th>CPU 1515 (F/T/TF)</th>
<th>CPU 1515 SP PC (F/T/TF)</th>
<th>CPU 1516 (F/T/TF)</th>
<th>CPU 1507S (F)</th>
<th>CPU 1517 (F/T/TF)</th>
<th>CPU 1518 (F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of imported companion specifications (information models)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of OPC UA server interfaces</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of OPC UA nodes in user-defined server interfaces</td>
<td>1000</td>
<td>5000</td>
<td>30000</td>
<td>30000</td>
<td>30000</td>
<td>30000</td>
<td>30000</td>
<td>30000</td>
<td>30000</td>
</tr>
<tr>
<td>Maximum size of loadable OPC UA server interfaces</td>
<td>1024 KB</td>
<td>5120 KB</td>
<td>15360 KB</td>
<td>15360 KB</td>
<td>15360 KB</td>
<td>15360 KB</td>
<td>15360 KB</td>
<td>15360 KB</td>
<td>15360 KB</td>
</tr>
<tr>
<td>Provision of methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of usable server methods or max. number of server method instances (instructions OPC_UA_ServerMethodPre, OPC_UA_ServerMethodPost)</td>
<td>20</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
9.4 Using the S7-1500 CPU as an OPC UA client

9.4.1 Overview and requirements

With STEP 7 (TIA Portal) Version 15.1 and higher, you can configure and program an OPC UA client that can read PLC tags in an OPC UA server. Furthermore, it is possible to transfer new values for PLC tags to an OPC UA server. In addition, you can call methods that an OPC UA server provides in your user program. You use the new instructions for OPC UA clients for this purpose in your user program.

The instructions of the OPC UA client are based on the standard "PLCopen OPC UA Client for IEC61131-3".

PLCopen companion specification

With these standardized instructions, you can use OPC UA client functions in your user program that can be executed in an S7-1500 CPU.

In addition, it is possible with just a few adaptations to run this user program in controllers of other manufacturers if these manufacturers also implemented the OPC UA Companion Specification "PLCopen OPC UA client for IEC61131-3".

Convenient editors in STEP 7

For the parameter assignment of the instructions for OPC UA clients, a convenient editor is available in the TIA Portal – the connection parameter assignment (Page 245).

As of Version 15.1, STEP 7 also features an editor for client interfaces (Page 228).

This section describes how you work with these editors.

First, you will be shown how to create and configure a new interface with the interface editor, because you need this type of interface for the subsequent connection parameter assignment.

The description uses an example for better comprehensibility, see Description of the example (Page 227).

Requirement

To use the client of the S7-1500 CPU, you must enable it:

1. Select the area "OPC UA > Client" in the properties of the CPU.
2. Select the "Enable OPC UA client" option.

If you do not enable the client, the connection is not established. You receive a corresponding error message at the instructions, for example "OPC_UA_Connect".
Using the S7-1500 CPU as an OPC UA client

Overview

To use the editor and the connection parameter assignment, follow these steps:

1. First, specify a client interface. Add to the interface the PLC tags and PLC methods that you want to access ("First step (Page 228)").
2. Next, configure the connection to the OPC UA server ("Second step (Page 245)").
3. Finally, use the configured connection for the OPC UA client instructions ("Third step (Page 253)").

9.4.2 Useful information about the client instructions

With the standardized OPC UA client instructions you are able to control communication for the following tasks with the S7-1500 CPU as an OPC UA client:

- Read/write tags of the OPC UA server
- Call methods in the OPC UA server

Optional instructions can be used to determine the status of the connection between the OPC UA client and OPC UA server or to determine nodes along a known hierarchy.

Standardized sequence of OPC UA communication

The sequence of the communication, and thus the order of the instructions, follows a pattern that is illustrated in the following.

Run sequence for a read or write operation

1. Instructions for preparation of read and write operations
2. Read and write instructions
3. Instructions for “clean-up” after a completed read or write operation
Run sequence for a method call in the OPC UA server:

1. Instructions for preparation of method calls
2. Method calls
3. Instructions for "clean-up" after completed method calls

Optional instructions for reading out the status of a connection between the OPC UA client and OPC UA server and for reading out complete paths from the OPC UA server:

1. Instructions for preparation of read and write operations with inserted instruction for requesting, for example, the NodeIDs of nodes of the OPC UA server.
2. You can determine the connection status between the establishment and termination of the connection in parallel with other instructions.
3. Instructions for "clean-up"

**Convenient editors in STEP 7**

The OPC UA client instructions are described in detail in the reference part (STEP 7 information system). For parameter assignment of the instructions, a convenient editor is available in the TIA Portal – the connection parameter assignment (Page 245).

We recommend starting with the connection parameter assignment for the first program draft and using additional instructions and manually optimizing the program as required.

**Information about the client instructions**

The client instructions are described in detail in the help to the Instructions > Communication > OPC UA > OPC UA client.
9.4.3 Number of client instructions that can be used simultaneously

SIMATIC error codes for OPC UA client instructions

The following limits apply to the simultaneous use of OPC UA client instructions:

<table>
<thead>
<tr>
<th>OPC UA instruction</th>
<th>Maximum number for CPU 1510SP (F)</th>
<th>Maximum number for CPU 1505 (S/SP/SP F/SP T/SP TF)</th>
<th>Maximum number for CPU 1516 (F/T/TF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC_UA_Connect</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_NamespaceGetIndexList</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_NodeGetHandleList</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_MethodGetHandleList</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_TranslatePathList</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_ReadList</td>
<td>20 in total (max. 5 per connection, see OPC_UA_Connect)</td>
<td>50 in total (max. 5 per connection, see OPC_UA_Connect)</td>
<td>200 in total (max. 5 per connection, see OPC_UA_Connect)</td>
</tr>
<tr>
<td>OPC_UA_WriteList</td>
<td>20</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>OPC_UA_MethodCall</td>
<td>20</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>OPC_UA_NodeReleaseHandleList</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_MethodReleaseHandleList</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_Disconnect</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>OPC_UA_ConnectionGetStatus</td>
<td>4</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

Maximum number of usable OPC UA client interfaces

If you create OPC UA client interfaces using the connection parameter assignment, the number of client interfaces is limited to 40.

Create the OPC UA client interfaces by double-clicking the "Add new client interface" symbol in the project tree of the "OPC UA communication" area.

The maximum number of OPC UA client interfaces is independent of whether you also use the CPU as OPC UA server.
9.4.4 Example configuration for OPC UA

The following sections describe how you can use the client interfaces editor and the connection parameter assignment.

The description is based on a specific example: Two S7-1500 CPUs operate in the system: One CPU serves as the OPC UA client and the other as the OPC UA server.

You can, of course, also use controllers, sensors and IT systems of other manufacturers as OPC UA clients or servers. In particular, the data exchange between different systems (interoperability) is a major advantage of OPC UA.

Connection parameter assignment using an example:

The plant produces blanks in a production line.

The following controllers are used:

1. An S7-1511 CPU serves as the controller of the production line.
   
   The controller is named "Productionline" in the example.

   The OPC UA server of the controller is enabled.

   The CPU has the IP address 192.168.1.1 in the example.

   This CPU publishes the values of following tags via the OPC UA server:
   
   - **NewProduct**
     
     The tag has the data type "BOOL".
     
     When this PLC tag has the value TRUE, the production line has processed a blank.
     
     The blank is ready for pick-up.
   
   - **ProductNumber**
     
     This tag contains the identification number of the blank.
     
     The tag has the data type "Int".
   
   - **Temperature**
     
     This tag contains temperature values recorded during the production of the blank.
     
     The tag is an array with elements of the "Real" data type.

   In addition, this CPU provides the following writable tag:
   
   - **ProductionEnabled**
     
     This tag is set by the OPC UA client.
     
     The tag has the data type "BOOL".
     
     If the value is set to TRUE, the production line is released and may produce blanks.

   In addition, this CPU provides the following method via the OPC UA server:
   
   - **OpenDoor**
     
     OPC UA clients can hereby arrange for an access door to be opened to the production line.
2. An S7-1516 CPU controls the interaction with other production lines. This CPU is named "Supervisor" in the example. The OPC UA client of this CPU is enabled. Using OPC UA, this CPU can read the NewProduct and ProductNumber tags, set the ProductionEnabled tag and call the OpenDoor method. The CPU has the IP address 192.168.1.2 in the example.

The following figure shows the example in the network view of the TIA Portal:

9.4.5 Creating client interfaces

As of Version 15.1, the TIA Portal has an editor for client interfaces. You group all PLC tags that you want to read or write from an OPC UA server in a client interface. In addition, the client interface contains all methods that the OPC UA server provides and that you want to call with your user program (that acts as an OPC UA client). If you create a client interface, STEP 7 also creates data blocks for the parameter assignment of the connection to the OPC UA server from which you want to read data or to which you want to write data.

Maximum number of client interfaces

You can create a maximum of 40 client interfaces.
Editor for client interfaces

To create a client interface, follow these steps:

1. Select the project view in the TIA Portal.
2. In the "Devices" area, select the CPU you want to use as an OPC UA client.
3. Click "OPC UA communication > Client interfaces".
4. Double-click "Add new client interface".

STEP 7 creates a new client interface and display in the editor.

STEP 7 names the new interface "Client interface_1". If a "Client interface_1" already exists, the new interface receives the designation "Client interface_2" etc.

In addition, STEP 7 creates the following data blocks:

- **Client_Interface_1_Configuration**
  
  The data block already contains all system data types that are needed for the instructions of the OPC UA client.
  
  This data block is filled when you configure the connection to the OPC UA server.
  
  You configure a connection in the properties of the client interface, see: [Example configuration for OPC UA](Page 227).

- **Client_Interface_1_Data**
  
  A data block for the PLC tags that you want to read or write from an OPC UA server.
  
  You use this data block in your user program.
  
  This data block is currently still empty.

5. Select a descriptive name for the new client interface.

Select "Productionline" in the example.

This also changes the names of the associated data blocks to:

- **Productionline_Data**
- **Productionline_Configuration**
6. To import an OPC UA server interface, click the "Import interface" button in the top right of the editor.

   This allows you to import an XML file which describes the server interface of an OPC UA server.

   Alternative: To determine online the server interface of a connected OPC UA server, see:
   [Determine server interface online (Page 237)].

7. STEP 7 displays a dialog with which you can select an XML file.

   This XML file describes a server interface (address space of an OPC UA server).

   A server interface is the grouping of all PLC tags and server methods which an OPC UA server publishes.

   OPC UA clients can access the server interface:
   - Read PLC tags
   - Write PLC tags
   - Calling Server Methods

   For information on how to create a server interface for an OPC UA server, see [Create a server interface (Page 199)].

8. Create a read list in this client interface.

   To do this, follow these steps:
   - Click "Add new read list" in the left section of the editor.
     STEP 7 adds a new list named "ReadList_1".
     For the example, change the name to "ReadProduct"
   - Now add the new read list of the PLC tags that you want to read from the OPC UA server.
     In the example the "NewProduct" and "ProductNumber" tags are added to the "ReadProduct" read list.
     Select the "NewProduct" tag in the right-hand field of the editor ("OPC UA Server interface"). Drag the "NewProduct" tag to the "ReadProduct" read list in the middle field of the editor. Follow the same procedure with the "ProductNumber" tag.
The figure below shows the right field of the editor.

Alternative:
You can also select a new read list by dragging the right field of the editor ("OPC UA Server interface") to a node of the type Object or Folder and then dragging it to "Add new read list" in the left field of the editor. The new read list then contains all PLC tags of the node that has been moved.

In the example, select the object "Data_for_OPC_UA_Clients", which contains the tags "NewProduct" and "ProductNumber". STEP 7 generates the new write list "Data_for_OPC_UA_Clients". In addition, the object contains the tag "Temperature".

Delete the "Temperature" tag from the read list. The OPC UA server should not query the "Temperature" tag.

Change the name of the read list in "ReadProduct".

The following figure shows the content of the read list:
9. If you want assign new values to PLC tags, create a write list in this client interface.

   To do this, follow these steps:
   - Click "Add new write list" in the left section of the editor.
     STEP 7 adds a new list with the name "ReadList_1".
     For the example, change the name to "WriteStatus".
   - Now add the new write list of all OPC UA server tags to which you want to assign new values.
     In the example, add the "WriteStatus" tag to the write list "ProductionEnabled".
     Select the Tag of right field of the editor ("OPC UA Server interface"). Drag the tag to the write list in the middle field of the editor.

**Alternative:**

   You can also create a new write list by selecting a node of the type Object or Folder in the right field of the editor ("OPC UA server interface") and then dragging to "Add new write list" in the left field of the editor.

   The new write list then contains all tags of the relevant node.

   In the example, select the object "Data_from.OPC_UA_Clients", which contains the tag "ProductionEnabled". STEP 7 generates the new write list "Data_from.OPC_UA_Clients". Change the name in "WriteStatus".

   The following figure shows the content of the write list:

   ![WriteStatus Table]

10. If you want to call a method of this OPC UA server, generate a new method list.

   To do this, follow these steps:
   - In the left section of the editor, click "Add new method list".
     STEP 7 adds a new list with the name "Method list_1".
     For the example, change the name to "CallOpenDoor".
   - Now add a method of the OPC UA server to the new method list.
     In this example, add the method "OpenDoor" to the method list "CallOpenDoor".
     Select the method of right field of the editor ("OPC UA Server interface"). Drag the method to the method list in the middle field of the editor.
Alternative:

You can also generate a new method list by selecting a method (node of the type Object) in the right field of the editor (OPC UA Server interface) and then dragging it to "Add new method list" in the left field of the editor. The new method list then contains the method of the relevant node.

The following figure shows the content of the method list:

<table>
<thead>
<tr>
<th>CallOpenDoor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the node</td>
</tr>
<tr>
<td>Method</td>
</tr>
</tbody>
</table>

If you want to call another method of the OPC UA server, you must create a new method list. Each method list contains only one method.

11. Compile the project.

To do so, select the project and click the following button in the toolbar:

![Compile button]

STEP 7 compiles the project and updates the data blocks that belong to the "ProductionLine" client interface.

---

**Note**

During compilation, STEP 7 overwrites all data in the data blocks belonging to the client interface. For this reason, you should neither add to nor correct these data blocks manually.

---

**Note**

**Renaming nod names (DisplayNames)**

In read lists, write lists and method lists you can rename the name of a node by means of the shortcut menu. This is the "DisplayName" in the OPC UA language usage.

If you rename the name of a method list node and the node is already used in a programmed block for the method call "OPC_UA_MethodCall", the compilation of the project leads to consistency errors: During the compilation the UDTs of the method are generated with the changed name. The references to the method used in the program are then no longer correct.

To correct the consistency errors, you can either undo the name change of the method in the client interface or navigate to the method call and assign the relevant parameters again there under "Properties > Block parameters" ("Configuration" tab).  

---
Data blocks of client interface

The following data blocks belong to the "Productionline" client interface:

- **Productionline_Configuration**
  
  A data block for the configuration.

  In the example, this data block is called "Productionline_Configuration".

  The data block already contains all system data types that are needed for the instructions of the OPC UA client.

  In addition, the data block contains general default values for parameter assignment of the connection to an OPC UA server.

  If you are working with connection parameter assignment, this data block will be filled.

- **ProductionLine_Data**
  
  A data block for the PLC tags that you have entered in the client interface editor.

  In the example, this data block is called "Productionline_Data".

  The figure below shows the data block.

  ![Data Block Diagram](image)

  Use the "ProductionLine_Data" data block in your user program and access the read values of the "NewProduct" and "ProductNumber" PLC tags. This is explained in the following section using an example.
Reading and writing PLC tags of the client interface

Example: Reading the "ProductNumber" value

For example, you write in an SCL program:

```scl
#MyLocalVariable := "ProductionLine_Data".ReadProduct.Variable.ProductNumber;
```

You use this, for example, to assign the number of the blank that was just produced in the production line to the local tag "#MyLocalVariable".

Requirements:

- A connection exists to the OPC UA server of the CPU, which controls the production line.
- The OPC UA client has read the current values.

For this reason you check whether a read value is valid:

- Check whether the value in "ProductionLine_Data".Product.NodeStatusList[1]" is equal to 0.
- Check when this value was sent from the OPC UA server. This value is in "ProductionLine_Data".Product.TimeStamps[1].

Example: Writing the "ProductEnabled" value

Transfer the new values for PLC tags, in the example for the "ProductionEnabled" tag, to the OPC UA server using the data block.

With the following assignment, you enable the production line in the example plant:

```scl
"ProductionLine_Data".WriteStatus.Variable.ProductionEnabled := TRUE;
```

This is only successful, however, if the following requirements are met:

- A connection exists to the OPC UA server of the CPU, which controls the production line.
- Current values are being written via the OPC UA client
Consistency check

Finally, check the consistency of the read/write list or method list.

1. Select the list that you want to check.
2. Click the "Consistency check" button above the "OPC UA client interface" area.

A green check mark indicates an error-free assignment of the tags or methods to the corresponding elements of the server interface.

You can assume that the data exchange between client and server and method calls operate without problem in runtime.

In the event of an error a list appears in the Inspector window. From this list you can jump to the respective error.

During the consistency check, STEP 7 checks

- whether all elements that you use in the respective list are also present in the server.
- Do the data types used match?
- For methods: Do the number, name, order, and data types of method arguments match?
9.4.6 Determine server interface online

With STEP 7 (TIA Portal) you can determine the interface of an OPC UA server online. This provides information on which tags of a connected OPC UA server you can read or set (write) with OPC UA clients. It also provides information on which server methods of the OPC UA server are available for OPC UA clients.

If you are working offline you can create the interface of the OPC UA server by means of an OPC UA XML file. The address space of the server is described in the OPC UA XML file, see: Export OPC UA XML file (Page 167).

Determine online server interfaces

To determine a server interface online, follow these steps:

1. In the STEP 7 project tree, select the CPU which is configured as OPC UA client (Supervisor in the example).

2. Select the client interface (in the example, OPC UA Communication > Client interfaces > Productionline).

   If no client interface has been created, double-click "Add new client interface".

3. Double-click the selected client interface.

   The editor for client interfaces is displayed.

4. In the left section of the editor, click "Add new read list", "Add new write list", or "Add new method list".

5. In the right field of the editor, select "Online[]" as data source for "Source of server data":

   Online []
6. Click the "Online access" button.

STEP 7 displays the "Connect to OPC UA server" dialog:

```
Access to the OPC UA server
OPC UA server address: opc.tcp://
Session name: um:USER:PC:Siemens:TIA:Portal@User:Client interface_1

Select OPC UA server end point:
Application name / security policy

TIA Portal certificate
Certificate location: None
Certificate (client): ...

User authentication
User authentication: Guest
User name: 
Password: 

Online status information:
```

Tip: When you are establishing an online connection to an OPC UA server for the first time, use the "Online access" button. When reconnecting after a disconnection, select the "Connect to online server" button next to the "Online" selection field.

In the top right, enter the IP address of the OPC UA server whose server interface you want to determine online (in the example, 192.168.1.1).

7. Click "Find selected server".

STEP 7 establishes a connection to the OPC UA server and determines all security settings (server endpoints) which the server provides.

STEP 7 displays the end points as list:

```
- SIMATIC S7-1500 OPC-UA Application: Production online
  - No Security
  - Basic128Rs15 - Sign
  - Basic128Rs15 - Sign & Encrypt
  - Basic256 - Sign
  - Basic256 - Sign & Encrypt
  - Basic256Rs236 - Sign
```

8. Click on the end point you want to use for a connection of step 7 to the OPC UA server.
9. Do you want to use a secure connection?
   - If you have selected a secure end point, then select the entry "TIA Portal" for the "Certificate location".
     And under "Certificate (Client)", select a client certificate for your PC on which STEP 7 (TIA Portal) is currently running.
     If a client certificate does not yet exist for your PC, you can generate a client certificate here in the TIA Portal.
     Proceed as follows to generate a certificate for your PC:
       - Click on the button in the "Certificate (Client)" input field.
       - Click "Add".
       - For "Common name of subject", enter "STEP 7 (TIA Portal)".
       - Select the "OPC UA client" entry at "Usage".
       - For "Subject Alternative Name (SAN)", enter the IP address of your PC on which you are currently running STEP 7 (TIA Portal) under "Value". Overwrite the already entered IP address.
       - If your PC uses a second IP address, enter this address as well. If your PC does not use a second IP address, delete the second IP address already entered.
       - Click "OK".
     - If you have not selected a secure end point, keep the default setting ("None").

10. How do you want log on?
    - If you want to log onto the OPC UA server as guest, then apply the default with "User authentication".
    - If you want to log on with user name and password, select "User name and password".
      Use the user name and password which was stored during the configuration of the OPC UA server in the properties of the CPU under "General > OPC UA > Server > Security > User authentication > User management".

11. Click "Connect".
    When a secure connection is established, a message appears that you must accept the server certificate for the secure connection to be established. In the message window, you can display further details about the server certificate via a link.
    This standard Windows window only provides information about the server certificate. If you click on the button to install the server certificate, the server certificate is not included in the certificate memory of the TIA Portal, i.e. the next time you connect you will be prompted to accept the server certificate again.
    STEP 7 then establishes a connection to the OPC UA server and again displays the editor for client interfaces.
    In the right field of the editor, STEP 7 displays the uppermost level of the address space of the OPC UA server:
12. Click on the small black triangle next to "Objects". 
   STEP 7 now also displays the level below Objects.

13. Click on the small black triangle next to "Productionline".
   STEP 7 now also displays the level below Productionline.

14. Now open additional lower-level folders:

   Alternative:

   To open the server interface in one stop, click the following symbol:

   ![Symbol]

   STEP 7 now displays the fully expanded server interface.

   If you click on this symbol again, STEP 7 again displays only the uppermost level of the server address space.
9.4.7 Using multilingual texts

In the client interface editor, you are also importing texts that can be displayed in different languages with the OPC UA XML files (information models). Multilingualism is optional, and each node can be defined differently regarding the languages it offers.

In the XML file, these are the following fields that can be prepared for different languages:

- DisplayName
- Description

Example for multilingual texts in an OPC UA XML file

In the XML file below, the display name and the description, for example, are entered with a "default" text and multiple localizable texts.

- Default text is the first entry without localization information.
- Localized text is the text after "Locale=" followed by a language code, e.g. "it-IT" for Italian.

```
<UAVariable NodeId="ns=3;i=6070" BrowseName="3:EngineeringRevision" ParentNodeId="ns=3;i=1002"
DataType="String">
  <DisplayName>EngineeringRevision</DisplayName>
  <DisplayName Locale="en-US">EngineeringRevision</DisplayName>
  <DisplayName Locale="de-DE">Revisionssstand</DisplayName>
  <Description>Revision Level of the engineering environment.</Description>
  <Description Locale="en-US">Revision Level of the engineering environment.</Description>
  <Description Locale="de-DE">Revisionssstand der Engineeringumgebung.</Description>
  <Description Locale="fr-FR">Niveau de révision de l'environnement d'ingénierie.</Description>
  <Description Locale="it-IT">Livello di revisione dell'ambiente di ingegneria.</Description>
</UAVariable>
```
Display of multilingual texts

When importing a server interface, the available multilingual texts are saved internally and downloaded to a CPU together with the project.

The client editor displays the text from the OPC UA XML file in the columns "Name of the node" (corresponds to "DisplayName") and "Description" (corresponds to "Description").

The following cascading rules determine which language is shown for a node:

- When the node contains text in the currently used editing language, the text is also displayed in the editing language.
  (Setting the editing language: In the project tree, select the area "Languages & resources > Project languages")

- When the node does not contain text in the editing language but a default text is defined there (without language code), the default text is displayed.

- "Name of the node" column: If no default text is defined either but a text in any other language exists, the DisplayName text is displayed in the first available language. This rule does not apply to description texts (Description).

- If none of the conditions listed above is met, no text is displayed.

When you change the editing language, the multilingual text in the imported interface will also change according to the rules explained above.

You can then apply the nodes in the corresponding lists (read list, write list, method list) with drag and drop.

You cannot change the language in the lists (read list, write list, method list).

Applying the displayed description texts as comment in PLC data types

When you compile the program, STEP 7 automatically creates PLC data types (UDTs) for each read list, for each write list and for inputs or outputs of each method. These UDTs each have one element for each node.

The UDTs apply the description text as comment according to the rules stated above. STEP 7 creates the comment in only one language, just like the texts in the OPC UA server interface can only be displayed in one language.
9.4.8 Rules for the access to structures

The rules for the access to structures are explained below. Note these rules when reading and writing values of complete structures provided by an OPC UA server.

How the client of the S7-1500 CPU accesses structures

The OPC UA client of the S7-1500 CPU uses neither TypeDictionaries nor DataTypeDefinition attributes, which a server offers for the resolution of these structures.

These options of the OPC UA client for checking structural elements in runtime are limited in the client.

Rules for the access to structures

If you use the client interfaces to configure the read and write lists (connection parameterization) and assign the PLC data types to the imported or online determined address model of the server, the read and write accesses to structures operate trouble-free in runtime.

The configuration by means of client interface automatically ensures that the sequence and the data type of the structural elements are coordinated on client and server side.

In runtime the OPC UA client only checks the total length of the transmitted value; more detailed checks are not possible.

Mapping rules apply to the assignment of OPC UA structures to PLC tags or DB tags (see AUTOHOTSPOT). Non-executed data type (such as OPC UA byte strings) are not supported.

Example of an error-free assignment of the structure elements

In the imported node set file (XML export), the structure is defined as follows:

```
<opcUeStruct>
  <bool>
    <myOPCUeStruct1> myOPCUeStruct
      <varA> Int32 </varA>
      <varB> Byte </varB>
      <nestedStructZ "myOPCUeStruct":"nestedStructZ">
        <varO> Float </varO>
        <varE> Double </varE>
        <varC> Double </varC>
    </nestedStructZ>
  </myOPCUeStruct1>
</bool>
```

The structure mapped in the read list matches, both in the order and in the assigned data types, the corresponding nodes of the node set file.

```
<myOPCUeStruct1>
  <varA> LINT </varA>
  <varB> US INT </varB>
  <nestedStructZ "myOPCUeStruct":"nestedStructZ">
    <varO> REAL </varO>
    <varE> LREAL </varE>
    <varC> LREAL </varC>
  </nestedStructZ>
</myOPCUeStruct1>
```
If the structure now changes on the server, for example tagA and tagB are swapped, and the read list remains the same in the client, the assignment is no longer correct:

- The total length of the data remains the same (only the order has changed)
- The configuration of the structure is different for client and server.

<table>
<thead>
<tr>
<th>WARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No error message in the case of different structure configuration between client and server</strong></td>
</tr>
</tbody>
</table>

If the structures of client and server do not match, this rule violation will possibly not generate any error during compilation and also not in runtime.

Make sure not to change the configured assignments for structures in runtime. If required, reconfigure the assignment in the read and write lists.
9.4.9 Using connection parameter assignment

9.4.9.1 Creating and configuring connections

With the instructions for OPC UA clients, you create a user program that exchanges data with an OPC UA server. A series of system data types are required for this.

To simplify your work with these system data types, a connection parameter assignment for OPC UA clients is available starting in STEP 7 (TIA Portal) Version 15.1.

Use of the connection parameter assignment is optional and not mandatory. You can also manually create the required system data types.

We use an example to make the description easier to follow, see description of the example (Page 227).

Opening the connection parameter assignment

To configure the connection to an OPC UA server, follow these steps:

1. In the "OPC UA communication" area, double-click the client interface whose parameters you want to assign in the project tree.

   For the example configuration: Double-click the "ProductionLine" client interface.

   The section "Create client interface (Page 228)" describes how to create a client interface.

2. Click the "Properties" tab (Inspector window) if the tab is not already displayed.

   STEP 7 now displays the connection parameter assignment for the instructions of the OPC UA client.

   The "General" tab is open.

3. Click the "Configuration" tab.

Setting the connection parameters

You set the connection to the OPC UA server in the "Configuration" tab.

1. Select a descriptive name for the session. For the example, select the name "OPC UA Connection to ProductionLine".

2. In the "Address" field, enter the IP address of the OPC UA server to which your user program (that operates as an OPC UA client) is to establish a connection.

   In the example configuration, the CPU that controls the production line has the IP address "192.168.1.1". A connection to the OPC UA server of this CPU is to be established. For this reason, you enter this IP address in the "Address" field.
3. If the OPC UA server is not using the standard port 4840, you must insert the port number here.
   For example, enter the number 48040 in the field, if the OPC UA server to which you want to establish a connection uses this port number.
   For the example, we accept the default setting "4840" because the OPC UA server of the example is accessible via port 4840.

4. Enter a path within the OPC UA server to restrict access to this path.
   When you specify a path, it is automatically entered at the "ServerEndpointUrl" entry in the configuration DB for the client interface. The entry then consists of the components "OPC Schematic Prefix", "IP address", "Port number" and "Server path", for example: "opc.tcp://192.168.0.10:4840/example/path". The information is optional. However, some servers only establish a connection if a server path is specified.

5. In addition, you accept the default settings for session timeout (30 seconds) and monitoring time (5 seconds).
   The following figure shows the connection parameters of the example:

```
<table>
<thead>
<tr>
<th>Connection parameter</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session name:</td>
<td>OPC UA Session to ProductionLine</td>
<td></td>
</tr>
<tr>
<td>Device:</td>
<td>Supervisor [CPU 1518-4 PN/DP MFP]</td>
<td>Unspecified device</td>
</tr>
<tr>
<td>Address:</td>
<td></td>
<td>192.168.178.151</td>
</tr>
<tr>
<td>Port:</td>
<td></td>
<td>4840</td>
</tr>
<tr>
<td>Path (optional):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server address:</td>
<td></td>
<td>opc.tcp://192.168.178.151:4840</td>
</tr>
<tr>
<td>Session timeout:</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>Monitoring time:</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
```
Setting the security parameters

1. Click the "Security" area in the "Configuration" tab.
   This area contains all security settings for the connection to the OPC UA server.

   The following settings are possible:

   "General" area

   Security mode:
   Select the security mode that the connection to the OPC UA server must meet from the drop-down list.
   If the server does not meet the selected mode, a session is not established.

   The following settings are available:
   • No security: No secure connection.
   • Sign: OPC UA server and OPC UA client sign the data transmission (all messages): Manipulations can thus be detected.
   • Sign & Encrypt: OPC UA server and OPC UA client sign and encrypt the data transmission (all messages):

   Security policy:
   Set the encryption techniques for the signing and encryption of messages.

   The following settings are possible:
   • No security
   • Basic128Rsa15
   • Basic256
   • Basic256Sha256

   To configure a secure connection, you must observe the following items:
   • A certificate is required for the client for a secure connection.
   • You have to make the client certificate known to the server.

   To find out how to proceed, see the section "Handling of the server and client certificates" under "Certificate of the OPC UA clients".

   "Certificates" area

   Client certificate:
   The certificate confirms the authenticity of the OPC UA client.

   To select a certificate, click the following symbol

   STEP 7 displays a list of certificates.
Select the certificate that you have introduced to the server, see the paragraphs under "Certificate of the OPC UA clients" in the section "Handling of the server and client certificates".

Click the symbol with the green check mark:

✅

Or, create a new certificate. To do so, click the "Add" symbol.

When you create a new certificate, you must introduce it to the server, see the paragraphs under "Certificate of the OPC UA clients" in the section "Handling of the server and client certificates".

"User authentication" area

The following settings are possible for user authentication:

- Guest
- User name and password
- Users (TIA Portal - Security Settings)

For more information, see AUTOHOTSPOT.

Setting languages

UA tags of the String type can be localized with OPC UA, that is, texts (values for the UA tag) can be available in different languages for the server. For example, localized texts can be available for DisplayName (Name of the node) and Description (Description).

In the "Languages" area of the "Configuration" tab you can, for example, influence the language of the texts returned by the server as follows:

In the "Languages" area, enter a number of languages that the server transfers to the client during connection setup.

The language or the local ID ("language code") associated with it that you enter in the first line is the language preferred by the client.

- If the server can provide the UA tag in the requested language, it is transferred to the client.
- If the server cannot provide the UA tag in the requested language, it checks whether it can provide the UA tag in the language you have entered in the second line (first substitute language).
- The server works its way down the list, and when it can provide neither the requested language nor a substitute language, it will provide the default language.
9.4.9.2 Handling of the client certificates of the S7-1500 CPU

Where does the client certificate come from?

If you are using the OPC UA client of an S7-1500 CPU (OPC UA client enabled), you can create certificates for these clients with STEP 7 V15.1 and higher as described in the following sections.

When you use UA clients from manufacturers or the OPC Foundation, a client certificate is generated automatically during installation or upon the first program call. You have to import these certificates with the global certificate manager in STEP 7 and use them for the respective CPU.

If you program an OPC UA client yourself, you can generate certificates through the program. Alternatively, you can generate certificates with tools, for example with OpenSSL or the certificate generator of the OPC Foundation:

- The procedure for OpenSSL is described here: "Generating PKI key pairs and certificates yourself (Page 149)."
- Working with the certificate generator of the OPC Foundation is described here: "Creating self-signed certificates (Page 148)."

Certificate of the OPC UA client of the S7-1500 CPU

A secure connection between the OPC UA server and an OPC UA client is only established if the server classifies the certificate of the client as trusted.

Therefore you have to make the client certificate known to the server.

The following sections describe how you can initially generate a certificate for the OPC UA client of the S7-1500 CPU and then make it available to the server.
1. Generate and export a certificate for the client

For a secure connection, you have to generate a client certificate and - if the server and client are located in different projects - export the certificate.

If client and server are in the same project, exporting the client certificate and subsequent import are not necessary.

Requirements

The IP interface of the CPU is configured, an IP address is available.

Background: The IP address under which the CPU can be accessed in your system is entered under "Subject Alternative Name (SAN)".

Generating an OPC UA client certificate

The easiest way to generate a client certificate for an S7-1500 CPU is to configure a client interface.

The configuration of the client interface provides for the selection or generation of a client certificate, see Creating and configuring connections (Page 245).

Alternatively, you can create the client as follows:

1. In the "Project tree" area, select the CPU you want to use as a client.
2. Double-click "Device configuration".
3. In the properties of the CPU click "Protection & Security > Certificate manager".
4. Double-click "<add>" in the "Device certificates" table.
   STEP 7 opens a dialog.
5. Click "Add".
6. Select the "OPC UA client" entry from the list at the "Usage".
7. Click "OK".
   STEP 7 now lists the client certificate in the "Device certificates" table.
8. If the server is in another project: Right-click this line and select "Export certificate" from the shortcut menu.
9. Select a directory in which you store the client certificate.
2. Announcing the client certificate to the server

You have to make the client certificate available to the server to allow a secure connection to be established.

To do so, proceed as follows:

1. If the client was configured in another project and you created and exported the client certificate there:
   - Select the "Use global security settings for certificate manager" option in the local certificate manager of the server. This makes the global certificate manager available.
     You will find this option under "Protection & Security > Certificate manager" in the properties of the CPU that is acting as server.
   - If the project is not yet protected, select "Security settings > Settings" in the STEP 7 project tree, click the "Protect this project" button and log on.
     The "Global security settings" item is now displayed under "Security settings" in the STEP 7 project tree.
   - Double click "Global security settings".
   - Double click "Certificate manager".
     STEP 7 opens the global certificate manager.
   - Click the "Device certificates" tab.
   - Right-click in the tab on a free area (not on a certificate).
   - Select the "Import" shortcut menu.
     The dialog for importing certificates is displayed.
   - Select the client certificate that the server is to trust.
   - Click "Open" to import the certificate.
     The certificate of the client is now contained in the global certificate manager. Note the ID of the client certificate just imported.

2. Click the "General" tab in the properties of the CPU that is acting as server.
3. Click "OPC UA > Server > Security > Secure Channel".
4. Scroll down in the "Secure Channel" dialog to the section "Trusted clients".
5. Double-click in the table on the empty row with "<add new>". A browse button is displayed in the row.
6. Click this button.
7. Select the prepared client certificate.
8. Click the button with the green check mark.
9. Compile the project.
10. Load the configuration into the S7-1500 CPU (server).
Result

The server now trusts the client. If the server certificate is also considered trusted, the server and client can establish a secure connection.

9.4.9.3 User authentication

In the OPC UA client interface of the S7-1500, you can set what authentication is required for a user of the OPC UA client wishing to access the server. To do so, you must select the corresponding client interface in the project tree of the requested S7-1500 CPU under "OPC UA communication > Client interfaces" and select the type of user authentication in the Inspector window under "Properties > Configuration > Security".

Types of user authentication

The following options are available for user authentication:

- **Guest**
  
The user does not need to verify authorization (anonymous access). The CPU creates an anonymous session for the user, and the OPC UA server does not check the authorization of the client user.

- **User name and password**
  
The user must prove authorization (no anonymous access). The OPC UA server checks whether the client user is authorized to access the server. Authorization is given by the user name and the correct password. These inputs cannot be checked by the client interface, which means all values are accepted as being valid.

  **Note**

  STEP 7 stores user name and password unencrypted in the data block/instance data block. Recommendation: Use the user authentication "User (TIA Portal - Security Settings)".

- **Users (TIA Portal - Security Settings)**
  
  You can enter a user name from the list of users entered in the project for authentication. The names of the registered users for the current project are available in the user administration in the project tree under "Security Settings > Users and roles". There you can also enter additional users.

  You can also enter a name that is not listed in the user administration of the project or leave the field blank. This is necessary when the corresponding user name is only provided by a different source during runtime, for example, via HMI or from a different OPC UA client.
"No Security" security policy and authentication via user name and password

You can set the following combination:

Security policy = "No Security" and authentication via user name and password.

- The OPC UA server of the S7-1500 supports this combination. OPC UA clients can connect and encrypt the authentication data or not.
- OPC UA client of the S7-1500 CPU also supports this combination: However, in runtime it only connects if it can send the authentication data encrypted via cable!

Result: With the following configuration, not connection can be established in runtime.

- S7-1500 as OPC UA client
- OPC UA server which supports no encryption of authentication data when "No Security" (="none") is set as security policy.

Additional information

See Users and roles with OPC UA function rights (Page 190)

9.4.9.4 Using a configured connection

Introduction

This section shows you how to use a configured connection for OPC UA instructions (third step).

Requirements

- You have created a client interface and added PLC tags and PLC methods to this interface, see ("First step (Page 228)").
- You have configured a connection to an OPC UA server (Second step (Page 245)).

Overview

To read data from an OPC UA server or write data to an OPC UA server, use the following instructions:

- OPC_UA_Connect
- OPC_UA_NamespaceGetIndexList
- OPC_UA_NodeGetHandleList
- OPC_UA_ReadList or OPC_UA_WriteList
- OPC_UA_NodeReleaseHandleList
- OPC_UA_Disconnect
Order of the OPC UA instructions

The following figure shows the order in which the OPC UA instructions are called in a user program in order to use these instructions to read or write PLC tags:

1. Instructions for preparation of read and write operations
2. Read and write instructions
3. Instructions for "clean-up" after a completed read or write operation

The "OPC_UA_NodeReleaseHandleList" instruction can be omitted if "OPC_UA_Disconnect" is called immediately afterwards.

STEP 7 (TIA Portal) automatically supplies the parameters of these instructions if you are using a client interface and a configured connection to an OPC UA server.

The procedure is shown in the following section.
Using a client interface and configured connection

To use a configured OPC UA connection, follow these steps:

1. Open your user program in the TIA Portal.

2. Using drag-and-drop, move the "UA_Connect" instruction into the program editor.
   You will find the instruction under "Instructions > Communication > OPC UA" in the TIA Portal.

3. Select a call option for the instruction
   The example uses a multi-instance.
   STEP 7 displays the instruction in the program editor.
   The editor for the Function Block Diagram (FBD) programming language uses the following display:

   ![Diagram of UA_Connect instruction]

   The editor for the Ladder Logic (LAD) programming language displays the instruction similarly.

4. Click the toolbox symbol in the editor for FBD or LAD.
   The symbol is located in the heading of the instruction:

   ![Toolbox symbol]

   If you are using the editor for STL or SCL: Click the small green rectangle below the first character of the instance name:

   `#OPC_UA_Connect_Instance`

   The example (Page 227) uses "#OPC_UA_Connect_Instance" as the instance name.
   STEP 7 displays the properties of the instruction in a separate dialog.
5. For "Client interface" select the client interface that you want to use for the instruction.

   We select the "ProductionLine" client interface in the example.

   STEP 7 now interconnects the "ProductionLine" client interface with the parameters of the OPC_UA_Connect instruction:

   "ProductionLine" is the interface that the OPC UA client of the example uses for data exchange with the OPC UA server "ProductionLine".

6. Using drag-and-drop, move the "UA_NamespaceGetIndexList" instruction into the program editor.

   You will find the instruction under "Instructions > Communication > OPC UA" in the TIA Portal.

   Select the "Multi-instance" call option.

   Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.

   Select the client interface that you want to use ("ProductionLine" in the example).

   STEP 7 now automatically interconnects all parameters of the "OPC_UA_NamespaceGetIndexList" instruction:
7. Using drag-and-drop, move the "UA_NodeGetHandleList" instruction into the program editor.

Select the "Multi-instance" call option.

Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.

Select the client interface that you want to use. The example uses the "ProductionLine" client interface.

Under "Data access > Read/Writelist" select the read list that you want to use (in the example the read list "Product").

STEP 7 now automatically interconnects all parameters of the "OPC_UA_NodeGetHandleList" instruction:
If you want to write data to an OPC UA server, select the write list you want to use under "Data access > Read/Writelist" (the "ProductionStatus" write list in the example).

8. Using drag-and-drop, move the "UA_ReadList" instruction into the program editor.
   Select the "Multi-instance" call option.
   Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.
   Select the client interface that you want to use. The example uses the "ProductionLine" client interface.
   Under "Data access > Read/Writelist" select the read list that you want to use (in the example the "Product" read list).
   STEP 7 now automatically interconnects all parameters of the "OPC_UA_ReadList" instruction.
   If you want to write data to an OPC UA server, use the "OPC_UA_Write" instruction and select the list of tags you want to send to the server under "Data access > Writelist" ("ProductionStatus" write list in the example).

9. If you use different read lists or write lists as program-controlled lists in your user program, move the "UA_NodeReleaseHandleList" instruction to the program editor using drag-and-drop operation.
   Select the client interface that you want to use.
   Now select a read list or write list that you want to release: Only release read or write lists that you rarely use, since re-registering is time-consuming.
   Then, repeat the steps starting with step 7 with the "UA_NodeGetHandleList" instruction.

10. Using drag-and-drop, move the "UA_Disconnect" instruction into the program editor.
    Select the "Multi-instance" call option.
    Click the toolbox symbol (LAD and FBD) or the small green box below the instance name (STL and SCL) if the editor is not already open.
    Select the client interface that you want to use. The example uses the "ProductionLine" client interface.
    STEP 7 now automatically interconnects all parameters of the "OPC_UA_Disconnect" instruction.
Supported instructions

For the following instructions, STEP 7 automatically supplies the parameters if you are using a client interface and a configured connection to an OPC UA server:

- OPC_UA_Connect
- OPC_UA_NamespaceGetIndexList
- OPC_UA_NodeGetHandleList
- OPC_UA_MethodGetHandleList
- OPC_UA_MethodReleaseHandleList
- OPC_UA_ReadList
- OPC_UA_WriteList
- OPC_UA_MethodCall
- OPC_UA_NodeReleaseHandleList
- OPC_UA_Disconnect
10.1 S7 routing

Definition of S7 routing

S7 routing is the transfer of data beyond S7 subnet boundaries. You can send information from a transmitter to a receiver across several S7 subnets. The gateway from one S7 subnet to one or more other subnets is provided by the S7 router. The S7 router is a device which has interfaces to the respective S7 subnets. S7 routing is possible via various S7 subnets (PROFINET/Industrial Ethernet and/or PROFIBUS).

Requirements for S7 routing

- All devices that can be reached in a network have been configured in a project in STEP 7 and downloaded.
- All devices involved in the S7 routing must receive routing information about the S7 subnets that can be reached through specific S7 routers. The devices obtain the routing information by downloading the hardware configuration to the CPUs, since the CPUs play the role of an S7 router.

In a topology with several consecutive S7 subnets, the following order must be kept to when downloading: First download the hardware configuration to the CPU(s) directly connected to the same S7 subnet as the PG/PC, then download one by one the CPUs of the S7 subnets beyond this starting with the nearest S7 subnet through to the S7 subnet furthest away.

- The PG/PC you want to use to establish a connection via a S7 router must be assigned to the S7 subnet it is physically connected to. You can assign the PG/PC to a PG/PC in STEP 7 under Online & Diagnostics > Online accesses > Connection to interface/subnet.
- For S7 subnets of the type PROFIBUS: Either the CPU must be configured as DP master or, if it is configured as a DP slave, the "Test, commissioning, routing" check box must be selected in the properties of the DP interface of the DP slave.
- S7 routing for HMI connections is possible as of STEP 7 V13 SP1.

Note

Firewall and S7 routing

A firewall does not recognize the IP address of the sender during S7 routing when the sender is located outside the S7 subnet adjacent to the firewall.

An overview of the devices that support the "S7 routing" function is provided in this FAQ [https://support.industry.siemens.com/cs/ww/en/view/584459].
S7 routing for online connections

With the PG/PC, you can reach devices beyond S7 subnets, for example to do the following:

- Download user programs
- Download a hardware configuration
- Execute test and diagnostics functions

In the following figure, CPU 1 is the S7 router between S7 subnet 1 and S7 subnet 2.

Figure 10-1  S7 routing: PROFINET - PROFINET
The following figure shows the access from a PG via PROFINET to PROFIBUS. CPU 1 is the S7 router between S7 subnet 1 and S7 subnet 2; CPU 2 is the S7 router between S7 subnet 2 and S7 subnet 3.

![S7 routing: PROFINET - PROFIBUS](image)

**Figure 10-2  S7 routing: PROFINET - PROFIBUS**

**S7 routing for HMI connections**

You have the option of setting up an S7 connection from an HMI to a CPU via different subnets (PROFIBUS and PROFINET or Industrial Ethernet). In the following figure, CPU 1 is the S7 router between S7 subnet 1 and S7 subnet 2.

![S7 routing via HMI connections](image)

**Figure 10-3  S7 routing via HMI connections**
S7 routing for CPU-CPU communication

You have the option of setting up an S7 connection from a CPU to another CPU via different subnets (PROFIBUS and PROFINET or Industrial Ethernet). The procedure is described based on examples in the section S7 communication (Page 114).

Using S7 routing

For the CPU, select the PG/PC interface and the S7 subnet in the "Go online" dialog of STEP 7. S7 routing is performed automatically.

Number of connections for S7 routing

The number of connections available for S7 routing in the S7 routers (CPUs, CMs or CPs) can be found in the technical specifications in the manuals of the relevant CPU/CM/CP.
S7 routing: Example of an application

The figure below shows the example of an application for remote maintenance of a system using a PG. The connection is made here beyond two S7 subnets via a modem connection.

You configure a remote connection via TeleService in STEP 7 using “Online access” or “Go online”.

Figure 10-5 Remote maintenance of a plant using TeleService

Additional information

- The allocation of connection resources with S7 routing is described in the section Allocation of connection resources (Page 271).
- You can find more information on setting up TeleService in the STEP 7 online help.
- You can find more information on S7 routing and TeleService adapters when you search the Internet using the following links:

See also

HMI communication (Page 65)
10.2 Data record routing

Definition of data record routing

Data can be sent over PROFINET from an engineering station to field devices via multiple networks. Since the engineering station addresses the field devices using standardized records and these records are routed via S7 devices, the term "data record routing" is used to refer to this type of routing.

The data sent using data record routing include the parameter assignments for the participating field devices (slaves) and device-specific information (e.g. setpoint values, limit values).

Data record routing is used, for example, when field devices of different manufacturers are used. The field devices are addressed using standardized data records (PROFINET) for configuration and diagnostics.

Data record routing with STEP 7

You can perform data routing with STEP 7 by calling a device tool (for example, PCT) via the TCI interface (Tool Calling Interface) and passing call parameters. The device tool uses the communication paths that STEP 7 also uses for communication with the field device.

No configuration is required for this type of routing except the installation of the TCI tools on the STEP 7 computer.
Example: Data record routing with the Port Configuration Tool (PCT)

You can use the Port Configuration Tool (PCT) to configure the IO link master of the ET200 and assign parameters to connected IO link devices. The subnets are connected via data record routers. Data record routers are, for example, CPUs, CPs, IMs, IO link master.

You can learn about the constellations of data record routers supported by the PCT in this FAQ [http://support.automation.siemens.com/WW/view/en/87611392].

The figure below shows an example configuration with the data record routing with PCT.

![Image of data record routing configuration with PCT](image-url)

Figure 10-6 Example configuration for data record routing with PCT

Additional information

- The differences that exist between "normal" routing and data record routing are described in this FAQ [https://support.industry.siemens.com/cs/ww/en/view/7000978].
- Whether or not the CPU, CP or CM you are using supports data record routing can be found in the relevant manuals.
- The allocation of connection resources with data record routing is described in the section Allocation of connection resources [Page 271].
- You can find additional information on configuration with STEP 7 in the STEP 7 online help.
11.1 Connection resources of a station

Introduction

Some communications services require connections. Connections occupy resources in the automation system (station). The connection resources are made available to the station by the CPUs, communications processors (CPs) and communications modules (CMs).
**Connection resources of a station**

The connection resources available depend on the CPUs, CPs and CMs being used and must not exceed a maximum number per station.

The maximum number of resources of a station is determined by the CPU. Each CPU has reserved connection resources for PG, HMI and Web server communication. This ensures, for example, that a PG can always establish at least one online connection with the CPU regardless of how many other communications services are already using connection resources.

In addition, dynamic resources exist. The difference between the maximum number of connection resources and the number of reserved connection resources is the maximum number of dynamic connection resources. The communications services PG communication, HMI communication, S7 communication, Open User Communication, Web communication and other communication (for example OPC UA) use the dynamic connection resources from the pool.

The figure below shows an example of how individual components make connection resources available to an S7-1500 station.

![Diagram showing connection resources of a station](image)

1. Available connection resources of the station, of which:
   - A  Reserved connection resources of the station
   - A + B  Connection resources of CPU 1518
   - C  Connection resources of communications module CM 1542-1
   - D  Connection resources of communications processor CP 1543-1

2. Maximum communications resources of the station using the example of a configuration from CPU 1518, CM 1542-1 and CP 1543-1

Figure 11-1  Connection resources of a station
### Number of connection resources of a station

Table 11-1  Maximum number of connection resources supported for some CPU types

<table>
<thead>
<tr>
<th>Connection resources of a station</th>
<th>1511</th>
<th>1512C</th>
<th>1513</th>
<th>1515</th>
<th>1516</th>
<th>1517</th>
<th>1518</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum connection resources of the station</td>
<td>96</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>320</td>
<td>384</td>
<td></td>
</tr>
<tr>
<td>of which reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>of which dynamic</td>
<td>86</td>
<td>118</td>
<td>182</td>
<td>246</td>
<td>310</td>
<td>374</td>
<td></td>
</tr>
<tr>
<td>Connection resources of the CPU</td>
<td>64</td>
<td>88</td>
<td>108</td>
<td>128</td>
<td>160</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>Max. additionally usable connection resources by plugging in CMs/CPs</td>
<td>32</td>
<td>40</td>
<td>84</td>
<td>128</td>
<td>160</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>Additional connection resources CM 1542-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Additional connection resources CP 1543-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>Additional connection resources CM 1542-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Additional connection resources CP 1542-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

The number of connection resources that a CPU or a communication module supports is specified in the device manuals in the Technical Specifications.
Example

You have configured a CPU 1518-4PN/DP with a communications module CM 1542-1 and a communications processor CP 1542-5.

- Maximum connection resources of the station: **384**
- Available connection resources:
  - CPU 1518-4 PN/DP: 192
  - CM 1542-1: 64
  - CP 1542-5: 16
  - Total: **272**

The setup provides 272 connection resources. By adding additional communications modules, the station can support a maximum of 112 additional connection resources.

Reserved connection resources

10 connection resources are reserved for stations with S7-1500 CPU, ET 200SP CPU and ET 200pro CPU based on S7-1500:

- 4 for PG communication required by STEP 7, for example, for test and diagnostics functions or downloading to the CPU
- 4 for HMI communication which are occupied by the first HMI connections configured in STEP 7
- 2 for communication with the Web server
11.2 Allocation of connection resources

Overview - occupation of connection resources

The following figure shows how different connections occupy the resources of the S7-1500.

① HMI communication: See below.
② Open User Communication: Connections of Open User Communication occupy a connection resource in every end point.
③ S7 communication: Connections of S7 communication occupy a connection resource in every end point.
④ Web communication: The Web server connection occupies at least one connection resource in the station. The number of occupied connections depends on the browser.
⑤ PG communication: The PG connection occupies one connection resource in the station.
⑥ OPC UA communication: Each session that the OPC UA server of the CPU establishes with an OPC UA client as a rule occupies one connection resource (other communication) in the station.

Figure 11-2 Allocation of connection resources
Connection resources

11.2 Allocation of connection resources

Connection resources for HMI communication

With HMI communication, the occupation of connection resources in the station depends on the HMI device being used.

Table 11-2 Maximum occupied connection resources for different HMI devices

<table>
<thead>
<tr>
<th>HMI device</th>
<th>Maximum occupied connection resources of the station per HMI connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Panel</td>
<td>1</td>
</tr>
<tr>
<td>Comfort Panel</td>
<td>2^1</td>
</tr>
<tr>
<td>RT Advanced</td>
<td>2^1</td>
</tr>
<tr>
<td>RT Professional</td>
<td>3</td>
</tr>
</tbody>
</table>

^1 If you do not use system diagnostics or alarm configuration, the station occupies only one connection resource per HMI connection.

Example: You have configured the following HMI connections for a CPU 1516-3 PN/DP:

- Two HMI connections to an HMI TP700 Comfort. (2 connection resources each)
- One HMI connection to an HMI KTP1000 Basic. (1 connection resource)

In total 5 connection resources are occupied for HMI communication in the CPU.
Connection resources for routing

To transfer data beyond S7 subnets ("S7 routing"), an S7 connection is established between two CPUs. The S7 subnets are connected via gateways known as S7 routers. CPUs, CMs and CPs in S7-1500 are S7 routers.

The following applies for a routed S7 connection:

- A routed connection occupies one connection resource each in both end points. STEP 7 shows these connection resources in the "Connection resources" table.

- On the S7 router, two special connection resources are occupied for S7 routing. STEP 7 does not show the special connection resources for S7 routing in the "Connection resources" table. The number of resources for S7 routing depends on the CPU. You will find the resources for S7 routing in the technical specifications of the CPU in "Number of S7 routing connections".

Data record routing also enables transfer of data beyond S7 subnets from an engineering station connected to PROFINET to various field devices via PROFIBUS.

With data record routing, as with S7 routing, two of the special connection resources for S7 routing are also occupied on every data record router.

**Note**

**Connection resources with data record routing**

With data record routing, on the data record router, two special connection resources for S7 routing are occupied. Neither the data record connection nor the allocated connection resources are displayed in the table of connection resources.
When are connection resources occupied?

The time for the occupation of connection resources depends on how the connection is set up (see section Setting up a connection (Page 30)).

- **Programmed setup of a connection:**
  As soon as an instruction to establish a connection is called in the user program (TSEND_C/TRCV_C or TCON), a connection resource is occupied.

  With suitable parameter assignment of the CONT parameter of the TSEND_C/TRCV_C instructions or by calling the TDISCON instruction, the connection can be terminated following data transfer and the connection resource is available again. When the connection is terminated, the connection resources on the CPU/CP/CM are available again.

- **Configured connections** (e.g. HMI connection):
  If you have configured a connection in STEP 7, the connection resource is occupied as soon as the hardware configuration is downloaded to the CPU.

  After using a configured connection for data transfer, the connection is not terminated. The connection resource is permanently occupied. To release the connection resource again, you need to delete the configured connection in STEP 7 and download the modified configuration to the CPU.

- **PG connection:**
  As soon as you have connected the PG to a CPU online in STEP 7, connection resources are occupied.

- **Web server:**
  As long as you have opened the Web server of the CPU in a browser, connection resources are occupied in the CPU.

- **OPC UA server:**
  A connection resource is occupied in the CPU as long as a session exists between the OPC UA server of the CPU and an OPC UA client.

Monitoring the maximum possible number of connection resources

**Offline**

During configuration of connections, STEP 7 monitors the occupation of the connection resources. If the maximum possible number of connection resources is exceeded, STEP 7 signals this with a suitable warning.

**Online**

The CPU monitors the use of connection resources in the automation system. If you establish more connections in the user program than those provided by the automation system, the CPU acknowledges the instruction to establish the connection with an error.

S7-1500 and S7-300 comparison

You will find a comparison of how the communication resources of the S7-1500 and S7-300 are managed in this FAQ [https://support.industry.siemens.com/cs/ww/en/view/109747092].
11.3 Display of the connection resources

Display of the connection resources in STEP 7 (offline view)

You can display the connection resources of an automation system in the hardware configuration. You will find the connection resources in the Inspector window in the properties of the CPU.

![Connection resources table]

Figure 11-4 Example: Reserved and available connection resources (offline view)

**Station-specific connection resources**

The columns of the station-specific connection resources provide information about the used and available connection resources of the station.

In the example, a maximum of 256 station-specific connection resources are available for the automation system.

- 10 reserved connection resources, of which 4 are already in use and a further 6 available. The used resources are divided up as follows:
  - 4 Resources for HMI communication

- 246 dynamic connection resources, of which 81 are already in use and a further 165 available. The used resources are divided up as follows:
  - 6 resources for HMI communication
  - 23 Resources for S7 communication
  - 52 resources for Open User Communication
The warning triangle in the column of the dynamic station resources is therefore displayed because the sum of the maximum available connection resources of CPU, CP and CM (= 294 connection resources) exceeds the station limit of 256.

**Note**

**Available connection resources exceeded**

STEP 7 signals the exceeding of the station-specific connection resources with a warning. To make full use of the connection resources from the CPU, CP and CM, either use a CPU with a higher maximum number of available station-specific connection resources or reduce the number of communications connections.

② **Module-specific connection resources**

The columns of the module-specific connection resources provide information about the use of resources on the CPUs, CPs and CMs of an automation system:

The display is per module and not per interface.

In the example, the CPU makes 128 connection resources available, of which 47 are already in use and a further 81 still available.

The used resources are divided up as follows:

- 6 resources for HMI communication
- 2 resources for S7 communication
- 39 resources for Open User Communication
Display of the connection resources in STEP 7 (online view)

If you are connected to the CPU online, you can also see how many resources are currently being used under "Connection information".

<table>
<thead>
<tr>
<th>Connection resources - online</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Station resources</strong></td>
</tr>
<tr>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td><strong>Maximum number of resources</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Maximum</strong></td>
</tr>
<tr>
<td>FG communication</td>
</tr>
<tr>
<td>HMI communication</td>
</tr>
<tr>
<td>S7 communication</td>
</tr>
<tr>
<td>Open user communication</td>
</tr>
<tr>
<td>Web communication</td>
</tr>
<tr>
<td>Other communication</td>
</tr>
<tr>
<td><strong>Total resources used</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Available resources</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Figure 11-5  Connection resources - online

The online view of the "Connection resources" table in addition to the offline view also contains columns with the connection resources currently being used. Thus, the online view displays all used connection resources in the automation system, regardless of how the connection was set up.

The "Other communication" row displays connection resources assigned for communication with external devices. The table is updated automatically.

**Note**

If a routed S7 connection goes through a CPU, the required connection resources of the CPU do not appear in the table of connection resources.
Display of the connection resources for HMI

For information regarding the availability and assignment of connection resources for HMI connections, refer to the "Connection resources" properties in the Inspector window of the offline view (in the context of the HMI device).

Figure 11-6  Connection resources - HMI communication

The following is displayed in the connection resources area:

- Number of available connections on the HMI reserved for HMI communication and HTTP communication
- Number of connection resources for HMI communication and HTTP communications used offline in the HMI
  
  If the maximum number of available connection resources for an HMI device is exceeded, a corresponding message is output by STEP 7.
- "Maximum number of used PLC resources per HMI connection". This parameter is a factor that is to be multiplied by the number of HMI connections used offline. The product is the number of HMI resources occupied on the CPU.

Displaying the connection resources in the Web server

You can display the connection resources not only in STEP 7, but also with a browser that displays the relevant page of the Web server.

12.1 Connection diagnostics

Connections table in the online view

After selecting a CPU in the Devices & networks editor of STEP 7, you will see the status of your connections displayed in the online view of the connections table.

![Connections table](image)

Figure 12-1 Online view of the connections table

After selecting the connection in the connections table, you obtain detailed diagnostic information in the "Connection information" tab.
"Connection information" tab: Connection details

![Connection details diagram]

Figure 12-2  Diagnostics of connections - connection details
"Connection information" tab: Address details

![Connection address details diagram]

Diagnostics via web server

You can evaluate diagnostic information from the CPU using a web browser via the integrated web server of a CPU.

On the "Communication" Web page, you will find the following information about communication via PROFINET in various tabs:

- Information on the PROFINET interfaces of the CPU (for example addresses, subnets, physical properties).
- Information on the quality of the data transfer (for example number of data packets sent/received error-free).
- Information about the allocation/availability of connection resources.
- The "Connection status" page is similar to the online view in STEP 7 and also provides an overview of all connections with detail view.
Diagnostics with the user program

When you program the T_DIAG instruction, you can evaluate diagnostic information about the configured and programmed connections of the CPU using the user program.

Additional information


12.2 Emergency address

If you cannot reach the CPU via the IP address, you can set a temporary emergency address (emergency IP) for the CPU. Via this emergency address, you can re-establish the connection with a CPU in order to load a device configuration with a valid IP address.

You can set an emergency address regardless of the protection level of the CPU.

When do you need an emergency address?

Your CPU cannot be reached in the following cases:

- The IP address of your PROFINET interface is assigned twice.
- The subnet mask is set incorrectly.

Requirements

- You have selected "Set IP address in the project" for the IP protocol in the device configuration in STEP 7.
- The CPU is in STOP mode.

Restoring a valid device configuration with an emergency address

1. Set the emergency address for the interface of the CPU with a DCP tool. For example, the SIMATIC Automation Tool has a DCP command "Define IP address". The maintenance LED of the CPU lights up. The diagnostic buffer also shows that an emergency address was activated for an Ethernet interface.

2. Load a STEP 7 project with a valid IP address into the CPU.

3. Switch the CPU off and on again. The emergency address is reset.

Result

The CPU starts up with the valid IP address.
Communication with the redundant system
S7-1500R/H

Introduction

Communication with the S7-1500R/H redundant system basically functions as with the S7-1500 standard system.

This chapter describes the special features and restrictions for communication with the S7-1500R/H redundant system.

Communication options for the S7-1500R/H redundant system

- Open User Communication via TCP/IP, UDP and ISO-on-TCP
- S7 communication as server
- HMI communication
- PG communication
- SNMP
- Time-of-day synchronization via NTP

Restrictions for communication with the S7-1500R/H redundant system

- Open User Communication:
  - no configured connections
  - Secure Open User Communication: Not supported as certificate management is not possible for the R/H CPUs:
    - If you have activated Secure OUC, then although you can compile the user program and load it, you cannot add certificates to the R/H-CPUs.
  - no FDL connections
  - Email: The S7-1500R/H CPUs with firmware version V2.6 do not support versions < V5.0 of the "TMAIL_C" instruction. Version V5.0 is not supported.
  - No support of connection descriptions according to "TCON_Param"
- no OPC UA
- no S7 communication as client
- no web server
- PG communication:
  - It is not possible to access two CPUs online at the same time. You can either access the primary CPU or the backup CPU.
  - Downloading blocks in redundant mode is not possible.
  - No support of the function "Upload device as new station"
- No S7-routing between the PROFINET interface X1 and the PROFINET-interface X2 of the CPUs
- The CPUs of the S7-1500R/H do not support any central communication modules
13.1 System IP addresses

The system IP address of the S7-1500R/H redundant system

In addition to the device IP addresses of the CPUs, the S7-1500R/H redundant system supports system IP addresses:

- System IP address for the PROFINET interfaces X1 of the two CPUs (system IP-address X1)
- System IP address for the PROFINET interfaces X2 of the two CPUs (system IP-address X2)

You use the system IP addresses for communication with other devices (for example, HMI devices, CPUs, PG/PC). The devices always communicate via the system IP address with the primary CPU of the redundant system. This ensures, for example, that the communication partner can communicate with the new primary CPU (previously backup CPU) in the RUN-Solo system state after failure of the original primary CPU in redundant operation.

There is a virtual MAC address for each system IP address.

You enable the system IP addresses in STEP 7.

Advantages of the system IP addresses compared to device IP addresses

- The communication partner communicates specifically with the primary CPU.
- Communication of the S7-1500R/H redundant system via a system IP address still also works in the event of the failure of the primary CPU.

Applications

You use the system IP addresses for the following applications:

- HMI communication with the S7-1500R/H redundant system: With an HMI you manage or monitor the process on the S7-1500R/H redundant system.
- Open User Communication with the S7-1500R/H redundant system:
  - Another CPU or an application on a PC accesses data of the S7-1500R/H redundant system.
  - The S7-1500R/H redundant system accesses a different device.

TCP, UDP and ISO-on-TCP-connections are possible.

Requirements

- The communication partner and the PROFINET interfaces of the two CPUs are located in the same subnet.
- The communication partner is connected to both CPUs, each via the same interface (e.g. X2).
- The system IP address is enabled.
Communication via the system IP address X2

If the CPUs of the S7-1500R/H redundant system have two PROFINET interfaces, preferably use the PROFINET interface X2 for communication with other devices.

The following figure shows a configuration in which the communication partners are connected via the respective PROFINET interfaces X2 with the CPUs of the redundant system S7-1500R/H.

Figure 13-1 Example: Communication of the S7-1515R redundant system via the system IP address X2

① Open User Communication between a different CPU and the S7-1500R/H redundant system
② HMI communication with the S7-1500R/H redundant system
③ Open User Communication between the S7-1500R/H redundant system and a PC
Communication with the redundant system S7-1500R/H

13.1 System IP addresses

Communication via the system IP address X1

The following diagram shows a configuration where the communication partners are connected with a switch to the PROFINET ring of the S7-1500R/H redundant system. The PROFINET ring connects the communication partners with the respective PROFINET interfaces X1 of the two CPUs.

As the CPU 1513R only has one PROFINET interface, connection via the PROFINET ring is the only possibility of communicating via the system IP address X1.

![Diagram showing communication via system IP address X1](image)

① Open User Communication between the S7-1500R/H redundant system and a different CPU
② HMI communication with the S7-1500R/H redundant system
③ Open User Communication between the S7-1500R/H redundant system and a PC

Figure 13-2 Example: Communication of the S7-1513R redundant system via the system IP address X1
Communication via the system IP addresses X1 and X2

If the CPUs of the redundant system S7-1500R/H have two PROFINET interfaces (X1 and X2), you can use the a system IP address for each PROFINET interface. PROFINET devices which are connected to the interfaces X1 of the CPUs communicate via the system IP address X1. PROFINET devices which are connected to the interfaces X2 of the CPUs communicate via the system IP address X2.

1. Open User Communication between the S7-1500R/H redundant system and a different CPU.
2. HMI communication with the S7-1500R/H redundant system
3. Open User Communication between the S7-1500R/H redundant system and a PC

Figure 13-3 Example: Communication of the S7-1515R redundant system via the system IP addresses X1 and X2
Enable system IP addresses

Requirements:
- STEP 7 V15.1 or higher
- redundant system S7-1500R/H with two CPUs, e.g. two CPUs 1513R-1PN

If the CPUs of the S7-1500R/H redundant system have two PROFINET interfaces (X1 and X2), then you can use a system IP address for both PROFINET interfaces. The following section describes how to enable the system IP address for the interface X1.

Proceed as follows to enable the system IP address for your S7-1500R/H redundant system:
1. In the network view of STEP 7, select the interface X1 of one of the two CPUs.
2. In the Inspector window go to "Properties" > "General" > "Ethernet addresses" in the area "System IP address for switched communication".
3. Select the check box "Enable the system IP address for switched communication".

STEP 7 automatically creates a system IP address.

4. Change the system IP address if necessary.
5. If required, change the virtual MAC address. To do this, in "Virtual MAC address", assign a project-wide unique value (value range 01H to FFH) for the last byte.

Note

Uniqueness of the virtual MAC address

The redundant system S7-1500R/H uses the Virtual Router Redundancy Protocol (VRRP) for the system IP address and associated virtual MAC address. If you use further devices with VRRP, e.g. switches, ensure the uniqueness of the MAC addresses within an Ethernet-Broadcast-Domain.

Result: The system IP address X1 for the PROFINET interface X1 of the two CPUs is enabled.
13.2 Response to Snycup

Response of communication connections via the system IP address in the system state SYNCUP

- HMI, PG- and S7-connections are temporarily closed. For a short time during the SYNCUP it is not possible to establish connections to the S7-1500R/H redundant system.
- All existing connections of Open User Communication are interrupted:
  - Connections set up by the CPUs of the redundant system as an active connection partner are set up again after the SYNCUP.
  - The S7-1500R/H redundant system sets up connection endpoints again for the passive connection establishment after the SYNCUP.
- The processing of running instances of the instructions TSEND and TRCV is stopped. The block parameter STATUS returns 80C4H (temporary communication error).

13.3 Response to primary-backup switchover

Response of communication connections via the system IP address during a primary-backup switchover

- Running instances of the instructions TSEND and TRCV are stopped and return the status 80C4H (temporary communication error).
- Connections successfully established by the S7-1500R/H redundant system are established again by the new primary CPU.
- The new primary CPU sets up connection endpoints again for the passive connection establishment.
13.4 Connection resources of the redundant system S7-1500R/H

Maximum number of connection resources of the S7-1500R/H redundant system

The S7-1500R/H redundant system supports a maximum number of connection resources. The CPU used defines the maximum number of resources for an S7-1500R/H station:

- CPU 1513R: max. 88 connection resources
- CPU 1515R: max. 108 connection resources
- CPU 1517H: max. 160 connection resources

Allocation of connection resources

Communication connections occupy communication resources in the S7-1500R/H redundant system.

Each communication connection to the S7-1500R/H redundant system occupies communication resources in the S7-1500R/H station. Depending on the IP address used, a communication connection also occupies resources in one or both CPUs of the S7-1500R/H redundant system.

The following table shows in which CPU a communication connection occupies connection resources depending on the IP address used.

<table>
<thead>
<tr>
<th>Connect via...</th>
<th>Connection resources of the station</th>
<th>Connection resources CPU with redundancy ID 1</th>
<th>Connection resources CPU with redundancy ID 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a system IP address</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>a device IP address of the CPU with redundancy ID 1</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>a device IP address of the CPU with redundancy ID 2</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>
Display of the occupied connection resources in STEP 7

Requirements: Online connection to the redundant system S7-1500R/H

You will find the online display of the connection resources in the inspector window under "Diagnostics" > "Connection information". STEP 7 always displays the connection resources of the selected CPU and the S7-1500R/H-station.

<table>
<thead>
<tr>
<th></th>
<th>Station resources</th>
<th>Module resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reserved</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Maximum number of resources</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td><strong>PG communication:</strong></td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td><strong>HMI communication:</strong></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>S7 communication:</strong></td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td><strong>Open user communication:</strong></td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td><strong>Web communication:</strong></td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td><strong>Other communication:</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Total resources used:</strong></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td><strong>Available resources:</strong></td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 13-5 Display of the connection resources of the S7-1500R/H redundant system in STEP 7
13.5 HMI communication with the redundant system S7-1500R/H

13.5.1 HMI connection via the system IP address

Requirements

- redundant system S7-1500R/H, e.g. CPU 1513R-1PN
- System IP address is enabled
- HMI device with PROFINETI-interface

Procedure

To set up a HMI connection to an S7-1500R/H redundant system, follow these steps:

1. In the network view of STEP 7, select a PROFINET interface of the HMI device.
2. Using a drag&drop operation, draw a line between the PROFINET interface of the HMI device and a PROFINET-interface of the S7-1500R/H redundant system. The HMI-device and the S7-1500R/H redundant system are networked together.

3. In the list of functions, click the "Connections" icon. This activates connection mode.
4. Using a drag-and-drop operation, draw a line between the HMI device and a CPU of the S7-1500R/H redundant system. The list "Connection partners" opens.
5. Select the S7-1500R/H redundant system in the list "Connection partners".

Result: You have set up a HMI connection between the HMI device and the S7-1500R/H redundant system. The HMI connection uses the system IP address. The HMI device always connects to the primary-CPU.
**Changing the HMI connection over to the device IP address**

To permanently change the HMI connection over to the selected CPU, clear the check box "Use the system IP address for switched communication" in the properties of the HMI connection. The HMI connection then uses the device IP address of the PROFINET interface. In the event of the failure of this CPU, then the HMI connection to this CPU permanently fails.

![Properties of the HMI connection](image)

**Note**

**Automatic setup of HMI connection**

When you drag-and-drop a tag from the S7-1500R/H redundant system into an HMI screen or into the HMI tag table, STEP 7 automatically sets up an HMI connection. This HMI connection exists by default between the PROFINET interface of the HMI device and the PROFINET interface X1 of the CPU with redundancy ID 1. The connection uses the device IP address of the PROFINET interface X1.

You can change the HMI connection to a system IP address in the properties of the HMI connection.
13.6 Open User Communication with the redundant system S7-1500R/H

The following table shows which protocols of the Open User Communication you can use for the S7-1500R/H redundant system and the matching system data types and instructions.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>System data type</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>TCON_QDN</td>
<td>Establish connection and send/receive data via: TSEND_C/TRCV_C, TCON, TSEND/TRCV or TCON, TUSEND/TURCV (connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td></td>
<td>TCON_IP_v4</td>
<td></td>
</tr>
<tr>
<td>ISO-on-TCP</td>
<td>TCON_IP_RFC</td>
<td>Establish connection and send/receive data via: TSEND_C/TRCV_C, TCON, TSEND/TRCV or TCON, TUSEND/TURCV (connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td>UDP</td>
<td>TCON_IP_v4</td>
<td>Establish connection and send/receive data via: TSEND_C/TRCV_C, TUSEND/TURCV/TRCV (connection can be terminated via TDISCON)</td>
</tr>
<tr>
<td></td>
<td>TADDR_Param</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TADDR_SEND_QDN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TADDR_RCV_IP</td>
<td></td>
</tr>
<tr>
<td>Modbus TCP</td>
<td>TCON_IP_v4</td>
<td>MB_CLIENT</td>
</tr>
<tr>
<td></td>
<td>TCON_QDN</td>
<td>MB_SERVER</td>
</tr>
</tbody>
</table>

Table 13-1 Protocols, system data types and usable instructions for Open User Communication with the redundant system S7-1500R/H
13.6.1 Setting up an Open User Communication connection via the system IP address

Introduction

The S7-1500R/H redundant system can communicate with other devices via Open User Communication.

You set up the connections in the user program, e.g. via the "TSEND_C" instruction. The S7-1500R/H redundant system does not support configured connections.

You can either set up the connections either via the device IP addresses or via the system IP addresses of the PROFINET interfaces.

Open User Communication via the system IP addresses of the S7-1500R/H redundant system

If you set up the connection via a system IP address, then communication always takes place via the primary CPU.

Recommendation: Always use a system IP address for Open User Communication.

Open User Communication via the device IP addresses of the S7-1500R/H redundant system

In redundant mode, the redundant system can establish or terminate connections and send or receive data via every device IP address.

If you set up the connection via a device IP address, then communication takes place via the associated CPU. In the event of the failure of the CPU, then the entire communication via the device IP addresses of this CPU fails.

Setting a connection via the system IP address

How to set up a connection from the S7-1500R/H redundant system to a different CPU via the system IP address is described below.

You set up the connection in the user program of the redundant system S7-1500R/H with a TSEND_C instruction. You create a corresponding TRCV_C instruction in the user program of the other CPU.

The procedure is described using the example of a TCP connection between the S7-1500R/H redundant system and a CPU 1516-3PN/DP.
**Requirements**

- S7-1500R redundant system with two CPUs 1513-1PN
- System IP address of the PROFINET interface X1 is enabled.
- CPU 1516-3PN/DP
- The PROFINET interfaces X1 of the CPUs 1513R and the PROFINET interface X2 of the CPU 1516-3PN/DP are located in the same subnet.

![Example configuration for TCP-connection](image)

**TSEND_C instruction in the user program of the S7-1500R/H redundant system**

To set up a TCP-connection to a different CPU, follow these steps:

1. Create a "TSEND_C" instruction in the user program.

![TSEND_C instruction](image)

2. Select the "TSEND_C" instruction.
3. In the Inspector window, go to "Properties" > "Configuration" > "Connection parameters". On the left-hand side you can see the S7-1500R/H redundant system as a local end point of the connection:
   - "Interface:" X1 is the preset interface.
   - "Subnet:" If the interface X1 is assigned to an S7-subnet, then STEP 7 displays the name of the S7-subnet.
   - The check box "Use address of the H-system" is selected. The system IP address of the S7-1500R/H redundant system is in "Address".

![Diagram of connection parameters](image)

Figure 13-11 S7-1500R/H: Assigning parameters to the TSEND_C instruction in STEP 7

4. In "Partners" under "End point:" select the CPU 1516-3PN/DP as the communication partner.
5. In "Partners" under "Interface:" select the PROFINET interface X2 of the CPU 1516-3PN/DP.
6. In "Local" under "Connection data" select the setting "<new>". STEP 7 creates a data block for the connection data in the user program of the S7-1500R/H redundant system.
7. In "Partners" under "Connection type" select the setting "TCP". STEP 7 creates a data block for the connection data in the user program of the other CPU.
TRCV_C instruction in the user program of the CPU 1516

Create a TRCV_C instruction in the user program of the CPU 1516-3PN/DP and assign parameters as below:

![Diagram showing TRCV_C instruction parameters](image)

Figure 13-12 S7-1500R/H: Assigning parameters to the TRCV_C instruction in STEP 7
Setting up a connection via a device IP address

To set up an OUC-connection via a device IP address of one of the two CPUs:

- Select a suitable PROFINET interface of the S7-1500R/H redundant system.
- Deselect the "Use address of H-system" check box.

Figure 13-13 OUC-connection via a device IP address

Reference

Industrial Ethernet Security with CP 1543-1

All-round protection - the task of Industrial Ethernet Security

With Industrial Ethernet Security, individual devices, automation cells or network segments of an Ethernet network can be protected. Data transfer can also be protected by a combination of different security measures:

- Data espionage
- Data manipulation
- Unauthorized access

Security measures

- Firewall
  - IP firewall with stateful packet inspection (layer 3 and 4)
  - Firewall also for Ethernet "non-IP" frames according to IEEE 802.3 (layer 2)
  - Bandwidth limitation
  - Global firewall rules

  All network nodes located in the internal network segment of a CP 1543-1 are protected by its firewall.

- Logging
  
  To allow monitoring, events can be stored in log files that can be read out using the configuration tool or can be sent automatically to a Syslog server.

- HTTPS
  
  For encrypted transfer of websites, for example during process control.

- FTPS (explicit mode)
  
  For encrypted transfer of files.

- Secure NTP
  
  For secure time-of-day synchronization and transmission.

- SNMPv3
  
  For secure transmission of network analysis information safe from eavesdropping.

- VPN groups
  
  You can combine the CP 1543-1 with other security modules into VPN groups through configuration. IPsec tunnels are established between all the security modules of a VPN group (VPN). All internal nodes of these security modules can communicate securely with each other through this tunnel.

- Protection for devices and network segments
  
  The firewall and VPN groups protective functions can be applied to the operation of single devices, multiple devices, or entire network segments.
Additional information
An overview with links to the most important contributions on Industrial Security is available in this FAQ [https://support.industry.siemens.com/cs/ww/en/view/92651441].

14.1 Firewall

Tasks of the firewall
The purpose of the firewall functionality is to protect networks and stations from outside influences and disturbances. This means that only certain previously specified communications relations are permitted.

To filter the data traffic, IPv4 addresses, IPv4 subnets, port numbers or MAC addresses among other things can be used.

The firewall functionality can be configured for the following protocol levels:
- IP firewall with stateful packet inspection (layer 3 and 4)
- Firewall also for Ethernet "non-IP" frames according to IEEE 802.3 (layer 2)

Firewall rules
Firewall rules describe which packets are permitted or forbidden in which direction.
14.2 Logging

Functionality

For test and monitoring purposes, the security module has diagnostics and logging functions.

- Diagnostics functions
  These include various system and status functions that you can use in online mode.

- Logging functions
  This involves the recording of system and security events. Depending on the event type, the recording is made in volatile or non-volatile local buffer areas of the CP 1543-1. As an alternative, it is also possible to record on a network server.
  The parameter assignment and evaluation of these functions is only possible with a network connection.

Recording events with logging functions

You specify which events should be recorded with the log settings. Here you can configure the following recording variants:

- Local logging
  With this variant, you record the events in local buffers of the CP 1543-1. In the online dialog of the Security Configuration Tool, you can then access these recordings, visualize them and archive them on the service station.

- Network Syslog
  With the network Syslog, you use a Syslog server in the network. This records the events according to the configuration in the log settings.

14.3 NTP client

Functionality

To check the time validity of a certificate and the time stamp of log entries, the date and time are maintained on the CP 1543-1 as on the CPU. This time can be synchronized with NTP. The CP 1543-1 forwards the synchronized time to the CPU via the backplane bus of the automation system. This way the CPU also receives a synchronized time for the time events in program execution.

The automatic setting and periodic synchronization of the time takes place either via a secure or non-secure NTP server. You can assign a maximum of 4 NTP servers to the CP 1543-1. A mixed configuration of non-secure and secure NTP servers is not possible.
14.4 SNMP

Functionality

Like the CPU, the CP 1543-1 supports the transfer of management information using the Simple Network Management Protocol (SNMP). To achieve this, an "SNMP agent" is installed on the CP/CPU that receives and responds to the SNMP queries. Information about the properties of devices capable of SNMP is contained in so-called MIB files (Management Information Base) for which the user needs to have the appropriate rights.

With SNMPv1, the "community string" is also sent. The "community string" is like a password that is sent along with the SNMP query. The requested information is sent when the "community string" is correct. The request is discarded when the string is incorrect.

With SNMPv3, data can be transferred encrypted. To do this, select either an authentication method or an authentication and encryption method.

Possible selection:

- Authentication algorithm: none, MD5, SHA-1
- Encryption algorithm: none, AES-128, DES

You can deactivate the use of SNMP for the CP/CPU. Deactivate SNMP if the security guidelines in your network do not permit SNMP or if you use your own SNMP solution.

To find out how to deactivate SNMP for the CPU, refer to section Disabling SNMP (Page 60).

14.5 VPN

Functionality

For security modules that protect the internal network, VPN (Virtual Private Network) tunnels provide a secure data connection through the non-secure external network.

The module uses the IPsec protocol (tunnel mode of IPsec) for tunneling.

In STEP 7 you can assign VPN groups to security modules. VPN tunnels are automatically established between all modules of a VPN group. A module in one project can belong to several different VPN groups at the same time in the process.
Glossary

Automation system
Programmable logic controller for the open-loop and closed-loop control of process chains of the process engineering industry and manufacturing technology. The automation system consists of different components and integrated system functions according to the automation task.

Backup CPU
If the R/H system is in RUN-Redundant system state, the primary CPU controls the process. The backup CPU processes the user program synchronously and can take over process control if the primary CPU fails.

Bus
Transmission medium that connects several devices together. Data transmission can be performed electrically or via optical fibers, either in series or in parallel.

Client
Device in a network that requests a service from another device in the network (server).

CM → Communications module

Communications module
Module for communications tasks used in an automation system as an interface expansion of the CPU (for example PROFIBUS) and providing additional communications options (PtP).

Communications processor
Module for expanded communications tasks covering special applications, for example in the area of security.

Consistent data
Data that belongs together in terms of content and must not be separated when transferred.

CP → Communications processor
CPU

Central Processing Unit - Central module of the S7 automation system with a control and arithmetic unit, memory, operating system and interface for programming device.

Device

Generic term for:

● Automation systems (PLC, PC, for example)
● Distributed I/O systems
● Field devices (for example, PLC, PC, hydraulic devices, pneumatic devices) and
● Active network components (for example, switches, routers)
● Gateways to PROFIBUS, AS interface or other fieldbus systems

Device certificates

Such certificates are signed by a certificate authority (CA).

The signature of an end-entity certificate is checked with the public key of the certificate authority certificate.

The "Subject" attribute must not be identical to the "Issuer" attribute.

The "Subject", for example, contains the name of a program as with the OPC UA application certificate.

"Issuer" is the certificate authority that signed the certificate.

The "CA" field must be set to "False".

DP master

Within PROFIBUS DP, a master in the distributed I/O that behaves according to the EN 50170 standard, Part 3.

→ See also DP slave

DP slave

Slave in the distributed I/O that is operated on PROFIBUS with the PROFIBUS DP protocol and behaves according to the EN 50170 standard, Part 3.

→ See also DP master
Duplex

Data transmission system; a distinction is made between full and half duplex.

Half duplex: One channel is available for alternate data exchange (sending or receiving alternately but not at the same time).

Full duplex: Two channels are available for simultaneous data exchange in both directions (simultaneous sending and receiving in both directions).

End-entity certificate

→ See also device certificate

Ethernet

International standard technology for local area networks (LAN) based on frames. It defines types of cables and signaling for the physical layer and packet formats and protocols for media access control.

Ethernet network adapter

Electronic circuitry for connecting a computer to an Ethernet network. It allows the exchange of data / communication within the network.

FETCH/WRITE

Server services using TCP/IP, ISO-on-TCP and ISO for access to system memory areas of S7 CPUs. Access (client function) is possible from a SIMATIC S5 or a third-party device/PC. FETCH: Read data directly; WRITE: Write data directly.

Field device

→ Device

Freeport

Freely programmable ASCII protocol; here for data transfer via a point-to-point connection.

FTP

File Transfer Protocol; a network protocol for transferring files via IP networks. FTP is used to download files from the server to the client or to upload files from the client to the server. FTP directories can also be created and read out and directories and files can be renamed or deleted.

HMI

Human Machine Interface, device for visualization and control of automation processes.
IE
→ *Industrial Ethernet*

IM
→ *Interface module*

**Industrial Ethernet**
Guideline for setting up an Ethernet network in an industrial environment. The essential difference compared with standard Ethernet is the mechanical ruggedness and immunity to noise of the individual components.

**Instruction**
The smallest self-contained unit of a user program characterized by its structure, function or purpose as a separate part of the user program. An instruction represents an operation procedure for the processor.

**Interface module**
Module in the distributed I/O system. The interface module connects the distributed I/O system via a fieldbus to the CPU (IO controller/DP master) and prepares the data for the I/O modules.

**Intermediate CA certificate**
This is a certificate authority certificate that is signed with the private key of a root certificate authority.
An intermediate certificate authority signs end-entity certificates with its private key.
The signature of these end-entity certificates is verified with the public key of the intermediate certificate authority.
The "Subject" and "Issuer" attributes of the intermediate CA certificate must not be identical. This certificate authority has after all not signed its certificate itself.
The "CA" field must be set to "True".

**IO controller, PROFINET IO controller**
Central device in a PROFINET system, usually a classic programmable logic controller or PC. The IO controller sets up connections to the IO devices, exchanges data with them, thus controls and monitors the system.
IO device, PROFINET IO device

Device in the distributed I/O of a PROFINET system that is monitored and controlled by an IO controller (for example distributed inputs/outputs, valve islands, frequency converters, switches).

IP address

Binary number that is used as a unique address in computer networks in conjunction with the Internet Protocol (IP). It makes these devices uniquely addressable and individually accessible. An IPv4 address can be evaluated using a binary subnet mask that results in a network part or a host part as a structure. The textual representation of an IPv4 address consists, for example, of 4 decimal numbers with the value range 0 to 255. The decimal numbers are separated by periods.

IPv4 subnet mask

Binary mask, with which an IPv4 address (as a binary number) is divided into a "network part" and a "host part".

ISO protocol

Communications protocol for message or packet-oriented transfer of data in an Ethernet network. This protocol is hardware-oriented, very fast and allows dynamic data lengths. The ISO protocol is suitable for medium to large volumes of data.

ISO-on-TCP protocol

Communications protocol capable of S7 routing for packet-oriented transfer of data in an Ethernet network; provides network addressing. The ISO-on-TCP protocol is suitable for medium and large volumes of data and allows dynamic data lengths.

Linear bus topology

Network topology characterized by the arrangement of the devices in a line (bus).

MAC address

Worldwide unique device identification for all Ethernet devices. The MAC address is assigned by the manufacturer and has a 3-byte vendor ID and 3-byte device ID as a consecutive number.

Master

Higher-level, active participant in the communication/on a PROFIBUS subnet. The master has rights to access the bus (token) and can request and send data.

→ See also DP master
Modbus RTU

Remote Terminal Unit; Open communications protocol for serial interfaces based on a master/slave architecture.

Modbus TCP

Transmission Control Protocol; Open communications protocol for Ethernet based on a master/slave architecture. The data are transmitted as TCP/IP packets.

Network

A network consists of one or more interconnected subnets with any number of devices. Several networks can exist alongside each other.

NTP

The Network Time Protocol (NTP) is a standard for synchronizing clocks in automation systems via Industrial Ethernet. NTP uses the connectionless UDP transport protocol for the Internet.

OPC UA

OPC Unified Automation is a protocol for communication between machines, developed by the OPC Foundation.

Operating states

Operating states describe the behavior of a single CPU at a specific time.

The CPUs of the SIMATIC standard systems have the STOP, STARTUP and RUN operating states.

The primary CPU of the redundant system S7-1500R/H has the operating states STOP, STARTUP, RUN, RUN-Syncup and RUN-Redundant. The backup CPU has the operating states STOP, SYNCUP and RUN-Redundant.

Operating system

Software that allows the use and operation of a computer. The operating system manages resources such as memory, input and output devices and controls the execution of programs.

PG

→ Programming device

PNO

→ PROFIBUS user organization
Point-to-point connection
Bidirectional data exchange via communications modules with a serial interface between two communications partners (and two only).

Port
Physical connector to connect devices to PROFINET. PROFINET interfaces have one or more ports.

Primary CPU
If the R/H system is in RUN-Redundant system state, the primary CPU controls the process. The backup CPU processes the user program synchronously and can take over process control if the primary CPU fails.

Process image (I/O)
The CPU transfers the values from the input and output modules to this memory area. At the start of the cyclic program, the CPU transfers the process image output as a signal state to the output modules. The CPU then reads the signal states of the input modules into the process image input. The CPU then executes the user program.

PROFIBUS
Process Field Bus - European Fieldbus standard.

PROFIBUS address
Unique identifier of a device connected to PROFIBUS. The PROFIBUS address is sent in the frame to address a device.

PROFIBUS device
Device with at least one PROFIBUS interface either electrical (for example RS-485) or optical (for example Polymer Optical Fiber).

PROFIBUS user organization
Technical committee dedicated to the definition and development of the PROFIBUS and PROFINET standard.

PROFIBUS DP
A PROFIBUS with DP protocol that complies with EN 50170. DP stands for distributed I/O = fast, real-time capable, cyclic data exchange. From the perspective of the user program, the distributed I/O is addressed in exactly the same way as the centralized IO.
PROFINET

Open component-based industrial communications system based on Ethernet for distributed automation systems. Communications technology promoted by the PROFIBUS user organization.

PROFINET device

Device that always has a PROFINET interface (electrical, optical, wireless).

PROFINET interface

Interface of a module capable of communication (for example CPU, CP) with one or more ports. A MAC address is assigned to the interface in the factory. Along with the IP address and the device name (from the individual configuration), this interface address ensures that the PROFINET device is identified uniquely in the network. The interface can be electrical, optical or wireless.

PROFINET IO

IO stands for input/output; distributed I/O (fast, cyclic data exchange with real-time capability). From the perspective of the user program, the distributed I/O is addressed in exactly the same way as the centralized IO.

PROFINET IO as the Ethernet-based automation standard of PROFIBUS & PROFINET International defines a cross-vendor communication, automation, and engineering model.

With PROFINET IO, a switching technology is used that allows all devices to access the network at any time. In this way, the network can be used much more efficiently through the simultaneous data transfer of several devices. Simultaneous sending and receiving is enabled via the full-duplex operation of Switched Ethernet.

PROFINET IO is based on switched Ethernet with full-duplex operation and a bandwidth of 100 Mbps.

Programming device

Programming devices are essentially compact and portable PCs which are suitable for industrial applications. They are identified by a special hardware and software configuration for programmable logic controllers.

Protocol

Agreement on the rules by which the communication between two or more communication partners transpires.

PtP

Point-to-Point, interface and/or transmission protocol for bidirectional data exchange between two (and only two) communications partners.
**Redundant systems**

Redundant systems have multiple (redundant) instances of key automation components. Process control is maintained if a redundant component fails.

**Ring topology**

All devices of a network are connected together in a ring.

**Root CA certificates**

→ *See also root certificate*

**Root certificate**

This is the certificate of a certificate authority: It signs end-entity certificates and intermediate CA certificates with its private key.

The "Subject" attribute and the "Issuer" of this certificate must be identical. This certificate authority has signed its certificate itself.

The "CA" field must be set to "True".

TIA Portal V14 has such a root CA certificate:

If you configure the OPC UA server of an S7-1500 in the TIA Portal, the TIA Portal generates an end-entity certificate for the OPC UA server and signs that certificate with its own private key.

The signature of this end-entity certificate can be verified with the public key of the TIA Portal. This key can be found in the root CA certificate of the TIA Portal.

**Router**

Network node with a unique identifier (name and address) that connects subnets together and allows transportation of data to uniquely identified communications nodes in the network.

**RS232, RS422 and RS485**

Standard for serial interfaces.

**RTU**

Modbus RTU (RTU: Remote Terminal Unit, transfers the data in binary form; allows a good data throughput. The data must be converted to a readable format before it can be evaluated.

**S7 routing**

Communication between S7 automation systems, S7 applications or PC stations in different S7 subnets via one or more network nodes functioning as S7 routers.
SDA service

Send Data with Acknowledge. SDA is an elementary service with which an initiator (for example DP master) can send a message to other devices and then receives acknowledgment of receipt immediately afterwards.

SDN service

Send Data with No Acknowledge. This service is used primarily to send data to multiple stations and the service therefore remains unacknowledged. Suitable for synchronization tasks and status messages.

Security

Generic term for all the measures taken to protect against

- Loss of confidentiality due to unauthorized access to data
- Loss of integrity due to manipulation of data
- Loss of availability due to the destruction of data

Self-signed certificates

These are certificates that you sign with your private key and use as end-entity certificates. The signature of these end-entity certificates is verified with your public key.

The "Subject" and "Issuer" attributes of self-signed certificates must be identical: You have signed your certificate yourself.

The "CA" field must be set to "False".

You can, for example, use self-signed certificates as application certificates for an OPC UA client.

The procedure required to generate a self-signed certificate with the certificate generator of the OPC Foundation is described here (Page ).

Server

A device or more generally an object that can provide certain services; the service is performed at the request of a client.

Slave

Distributed device in a fieldbus system that can only exchange data with a master after the master has requested this.

→ See also DP slave
SNMP

**Simple Network Management Protocol**, uses the wireless UDP transport protocol. SNMP works in much the same way as the client/server model. The SNMP manager monitors the network nodes. The SNMP agents collect the various network-specific information in the individual network nodes and makes this information available in a structured form in the MIB (Management Information Base). This information allows a network management system to run detailed network diagnostics.

Subnet

Part of a network whose parameters must be matched up on the devices (for example in PROFINET). A subnet includes the bus components and all connected stations. Subnets can be linked together, for example using gateways or routers to form one network.

Switch

Network components used to connect several terminal devices or network segments in a local network (LAN).

System states

The system states of the redundant system S7-1500R/H result from the operating states of the primary and backup CPUs. The term system state is used as a simplified expression that refers to the operating states that occur simultaneously on both CPUs. The redundant system S7-1500R/H has the system states STOP, STARTUP, RUN-Solo, SYNCUP and RUN-Redundant.

System IP address

In addition to the device IP addresses of the CPUs, the redundant system S7-1500R/H supports system IP addresses:

- System IP address for the X1 PROFINET interfaces of the two CPUs (system IP address X1)
- System IP address for the X2 PROFINET interfaces of the two CPUs (system IP address X2)

You use the system IP addresses for communication with other devices (for example, HMI devices, CPUs, PG/PC). The devices always communicate over the system IP address with the primary CPU of the redundant system. This ensures that the communication partner can communicate with the new primary CPU (previously backup CPU) in the RUN-Solo system state after failure of the original primary CPU in redundant operation.

TCP/IP

Time-of-day synchronization

Capability of transferring a standard system time from a single source to all devices in the system so that their clocks can be set according to the standard time.

Tree topology

Network topology characterized by a branched structure: Two or more bus nodes are connected to each bus node.

Twisted-pair

Fast Ethernet via twisted-pair cables is based on the IEEE 802.3u standard (100 Base-TX). The transmission medium is a shielded 2x2 twisted-pair cable with an impedance of 100 Ohms (22 AWG). The transmission characteristics of this cable must meet the requirements of category 5.

The maximum length of the connection between the terminal and the network component must not exceed 100 m. The connectors are designed according to the 100Base-TX standard with the RJ-45 connector system.

UDP

User Datagram Protocol; communications protocol for fast and uncomplicated data transfer, without acknowledgment. There are no error checking mechanisms as found in TCP/IP.

User program

In SIMATIC, a distinction is made between the CPU operating system and user programs. The user program contains all instructions, declarations and data by which a system or process can be controlled. The user program is assigned to a programmable module (for example, CPU, FM) and can be structured in smaller units.

USS

Universal Serial Interface protocol (Universelles Serielles Schnittstellen-Protokoll); defines an access method according to the master-slave principle for communication via a serial bus.

Web server

Software/communications service for data exchange via the Internet. The web server transfers the documents using standardized transmission protocols (HTTP, HTTPS) to a Web browser. Documents can be static or put together dynamically from different sources by the web server on request from the Web browser.
## Index

### A
- Advanced Encryption Algorithm, 40
- AES, 40
- Applicant, 43
- Asymmetric encryption, 41

### B
- BRCV, 115
- BSEND, 115

### C
- Certificate authorities, 43
- Certificate subject, 43
- CM, 17
- Communication
  - Data record routing, 265
  - HMI communication, 65
  - Open communication, 67
  - Open User Communication, 67
  - PG communication, 63
  - Point-to-point connection, 123
  - S7 communication, 114
  - S7 routing, 260
- Communication options
  - Overview, 22
- Communication via PUT/GET instruction
  - Creating and configuring a connection, 117
- Communications
  - Communication protocols, 68
  - Establishment and termination, 93
- Communications module, 17
- Communications processor, 17
- Communications services
  - Connection resources, 30
- Connection
  - Diagnostics, 279
  - Instructions for Open User Communication, 70
- Connection diagnostics, 279
- Connection resources
  - Data record routing, 273
  - Display in STEP 7, 275
  - Display in the Web server, 278
  - HMI communication, 272

### D
- Data consistency, 34
- Data record routing, 265
- Digital certificates, 43

### E
- E-mail, 22, 69, 89
- End-entity certificate, 46
- Establishment and termination of communications, 93
- Export file for OPC UA, 167

### F
- FDL, 69
- Fetch, 22
- Firewall, 301
- Freeport protocol, 123
- FTP, 22, 69, 89, 90

### G
- GET, 115

### H
- Handshake Protocol, 42
- HMI communication, 22, 65

### I
- IM, 21
- Industrial Ethernet Security, 300
- Interface module, 21
- Interfaces for communication, 18
Interfaces of communications modules
- Point-to-point connection, [20]
- Interfaces of communications processors, [19]
- IP address, emergency address (temporary), [282]
- ISO, [22, 68]
- ISO-on-TCP, [68, 77]

L
- Logging, [302]

M
- Man-in-the-middle attack, [43]
- Modbus protocol (RTU), [123]
- Modbus TCP, [69]

N
- NTP, [22, 302]

O
- Occupation of connection resources, [274]
- OPC UA
  - Certificate generator, [148]
  - DB tags, [163]
  - End points, [157]
  - Identifier, [133]
  - Introduction, [129, 130, 131]
  - Layer model, [153]
  - Namespace, [132]
  - OpenSSL, [149]
  - PLC tags, [163]
  - Secure channel, [152]
  - Secure connection, [152]
  - Security mechanisms, [143]
  - Security settings, [157]
  - Signing and encryption, [145]
  - X.509 certificates, [147]
- OPC UA client
  - Authentication, [252]
  - Basics, [138]
  - Certificate, [181, 249]
- OPC UA server
  - Address space, [134]
  - Addressing, [170]
  - Application name, [169]
  - Authentication, [189]
  - Basics, [155]
- Commissioning, [168]
- Customizing the server certificate, [183]
- Generating a server certificate, [175]
- Performance, [166]
- Performance increase, [166]
- Publishing interval, [173]
- Runtime licenses, [192]
- Sampling interval, [174]
- Security settings, [178, 186]
- Subscription, [172]
- TCP port, [172, 173]
- Write and read rights, [163]
- XML export file, [167]

Open communication
- Connection configuration, [77]
- Setting up e-mail, [89]
- Setting up FTP, [90]
- Setting up TCP, ISO-on-TCP, UDP, [77]

Open User Communication
- Features, [67]
- Instructions, [70]
- Protocols, [68]
- OpenSSL, [149]

P
- PCT, [266]
- PG communication, [22, 63]
- Point-to-point connection, [22, 123]
- Private Key, [38]
- Procedure 3964(R), [123]
- Protocols for Open User Communication, [68]
- Public Key, [38]
- PUT, [115]

R
- Record Protocol, [42]
- RFC 5280, [38]
- Root certificate, [46]

S
- S7 communication, [22, 114, 273]
- S7 routing, [260]
  - Connection resources, [273]
- Secure communication, [38]
- Secure Socket Layer, [42]
- Security, [300]
- Security measures, [300]
- Firewall, [301]
Logging, 302
NTP, 302
SNMP, 303
Self-signed certificates, 44
Server certificate, 183
Setting up a connection, 30
   By configuring, 81
   ISO connection with CP 1543-1, 82
Signature, 45
SNMP, 22, 303
SSL, 42
Symmetric encryption, 40
Syslog, 302
System data type, 72

T
TCON, 71
TCP, 22, 68, 77
TDISCON, 71
Time-of-day synchronization, 22
TLS, 42
Transport Layer Security, 42
TRCV, 71
TRCV_C, 71
TSEND, 71
TSEND_C, 71

U
UDP, 22, 68, 77
URCV, 115
USEND, 115
USS protocol, 123

W
Web server, 22
Write, 22

X
X.509, 38