OPC UA .NET Client for SIMATIC RFID devices

RFID / OPC UA / .NET / C#

https://support.industry.siemens.com/cs/ww/de/view/109769038
Legal information

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

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

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

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

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

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

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

Legal information ........................................................................................................ 2  
1 Introduction ........................................................................................................... 4  
  1.1 Overview ........................................................................................................ 4  
  1.2 Principle of operation .................................................................................. 5  
  1.3 Components used .......................................................................................... 7  
2 Engineering .......................................................................................................... 8  
  2.1 Programming the OPC UA Client Example................................................. 8  
     2.1.1 OPC UA RFID Client ........................................................................ 8  
     2.1.2 UAClientHelperAPI .......................................................................... 10  
     2.1.3 OPC UA .NET Stack ........................................................................... 12  
  2.2 Operation ....................................................................................................... 14  
     2.2.1 Commissioning OPC UA servers .......................................................... 14  
     2.2.2 Description of the user interface ......................................................... 14  
     2.2.3 Putting the OPC UA Client Example into Operation....................... 16  
     2.2.4 Establishing a connection to the OPC UA Server ............................ 16  
     2.2.5 Browsing the Address Space of the OPC UA Server ....................... 17  
     2.2.6 Subscribe to variables ....................................................................... 17  
     2.2.7 Subscribe to events ............................................................................ 18  
     2.2.8 Write/read variables .......................................................................... 21  
     2.2.9 Call methods ...................................................................................... 22  
3 Useful information ............................................................................................... 29  
  3.1 Certificate handling in the example program ............................................. 29  
  3.2 Information about the license model of the software ............................. 29  
4 Appendix ............................................................................................................. 30  
  4.1 Siemens services ......................................................................................... 30  
  4.2 Links and literature ..................................................................................... 31  
  4.3 Change documentation ............................................................................. 31
1 Introduction

1.1 Overview

OPC UA (Open Platform Communications Unified Architecture) is an M2M communication protocol specified by the OPC Foundation. The OPC specification was developed to provide an interoperable, secure and reliable communication protocol. OPC UA enables vertical and horizontal communication in the entire automation environment. Due to these characteristics, OPC UA is becoming more and more accepted as standard in the industrial environment.

The SIMATIC Ident products RF600 and RF18xC offer you an OPC UA server according to the OPC UA company specification "AutoID". This provides an additional option for direct process data exchange between the devices and many other systems that support OPC UA.

Content of the application example

This application example shows you how you can create a simple client under .NET to control or diagnose the SIMATIC Ident devices via OPC UA.

Figure 1-1

Advantages of the Application Example

This application example offers you the following advantages:

- A simple and extensible OPC UA client created in C# for the .NET runtime environment.
- A commented C# class that summarizes the basic OPC UA client functions and ensures faster implementation.
Basic knowledge

The following basic knowledge is required from the user:

- Basics of programming in C# / .NET
- Basics of OPC UA
- The basics of RFID (SIMATIC Ident)
- Basics in software security and certificate handling

1.2 **Principle of operation**

The following explains which components, functions and modes of operation are used in the application example.

General function description

The following figure shows the most important components of this application example:

Figure 1-2

A simple OPC UA .NET client for Windows PCs/PGs communicates with the OPC UA server of the SIMATIC Ident RF600 and RF18xC devices.

SIMATIC RF600 forms an independent UHF reader family. SIMATIC RF18xC describes a series of communication modules to which serial RFID readers (e.g. RF200 and RF300) are connected.

The OPC UA servers of the SIMATIC Ident devices are set up and configured via Web-Based Management (WBM). The OPC UA client is created in C# / .NET and internally uses the freely accessible OPC UA .NET stack of the OPC Foundation. The C# class "UAClientHelperAPI" is provided to simplify own implementations of a .NET client. This class summarizes the basic functions of the .NET stack of the OPC Foundation and simplifies the implementation of the basic functions considerably. Client and server are connected via Ethernet and communicate via OPC UA via TCP/IP.
The client example supports the following OPC UA service sets:

- Search and find servers: Discovery Service Set
- Create and end sessions: Sessions Service Set
- Navigate in the address space: View Service Set
- Read and write variables and attributes: Attribute Service Set
- Subscribe to variables and events: Subscription Service Set, MonitoredItem Service Set
- Call methods: Method Service Set

**Functional sequence**

The following function sequence applies to the client in this example:

Figure 1-3
1.3 Components used

The application example was created with these components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMATIC RF680R</td>
<td>1</td>
<td>6GT2811-6AA10-0AA0</td>
<td>Alternatively RF610R, RF615R, RF650R, RF685R; Firmware 2.0 or higher</td>
</tr>
<tr>
<td>SIMATIC RF188C</td>
<td>1</td>
<td>6GT2002-0JE40</td>
<td>Alternatively RF185C, RF186C, RF186CI, RF188CI;</td>
</tr>
<tr>
<td>Visual Studio 2017</td>
<td>1</td>
<td>-</td>
<td>Community version also possible</td>
</tr>
<tr>
<td>OPC UA .Net Stack</td>
<td>1</td>
<td>-</td>
<td>V1.04.353.0 Download: Reference (2)</td>
</tr>
</tbody>
</table>

**Note**
To run the program, the .NET Framework 4.7.2 must be installed on your PC/PG.

**Note**
Using the RF18xC communication module, you can operate RF readers of the RF200 and RF300 families as well as MV400 and MV500 via OPC UA.

**Note**
RF18xC offers parallel operation of OPC UA and PROFINET. The address space can only be accessed by OPC UA in a read-only mode.
RF600 can be operated either only via OPC UA or only via PROFINET.

This application example consists of the following components:

<table>
<thead>
<tr>
<th>Components</th>
<th>File name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project folder and EXE file</td>
<td>109769038 OPC_UA_RFID-Client_CODE_V1_1.zip</td>
<td>Visual Studio project with compiled executable program file</td>
</tr>
<tr>
<td>Documentation</td>
<td>109769038 OPC_UA_RFID-Client_DOC_V1_1_de.pdf</td>
<td>This document</td>
</tr>
</tbody>
</table>
2 Engineering

2.1 Programming the OPC UA Client Example

The following descriptions explain the functional principles and programming of the OPC UA client sample program.

2.1.1 OPC UA RFID Client

The OPC UA client example "RfidOpcUa" was created in C#/.NET.

Structural setup

The following figure shows the structural structure of the OPC UA Client example of this application example:

Figure 2-1

<table>
<thead>
<tr>
<th>EventItemFilterForm.cs</th>
<th>ScanStartForm.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScanFrom.cs</td>
<td>ScanStopForm.cs</td>
</tr>
<tr>
<td>ReadTagForm.cs</td>
<td>SetTagPasswordForm.cs</td>
</tr>
<tr>
<td>EditMonitoredItemForm.cs</td>
<td>SubscriptionForm.cs</td>
</tr>
<tr>
<td>KillTagForm.cs</td>
<td>WriteTagForm.cs</td>
</tr>
<tr>
<td>ConnectForm.cs</td>
<td>WriteValueForm.cs</td>
</tr>
<tr>
<td>AboutForm.cs</td>
<td></td>
</tr>
</tbody>
</table>

RfidForm.cs

UAClientHelperAPI.cs

UAGetRfidMethodIdentifiers.cs

OPC UA .NET SDK / Stack

Opc.Ua.Core.dll
Opc.Ua.Client.dll
BouncyCastle.Crypto.dll (Crypto-Bibliothek)

The class "UAClientHelperAPI" is a user-specific class that summarizes and preconfigures basic calls of the "OPC UA .NET Stack". Additionally, private methods are included to create and fill required objects for the "OPC UA .NET Stack". This class is extensible and reusable and can be used by developers to create their own simple OPC UA clients. The functionality of this class does not include all required OPC UA functions for this client, which is why the program has to access the "OPC UA .NET Stack" directly.
The "OPC UA .NET Stack" of the OPC Foundation contains the actual classes/objects which execute and manage the OPC UA communication. The stack consists of two libraries (DLLs):

- "Opc.Ua.Core.dll"
- "Opc.Ua.Client.dll"

For the cryptography required for OPC UA, the stack accesses the free software library "BouncyCastle.Crypto.dll".

You can find the download of the complete .NET stack as well as its documentation in the bibliography (21). This stack is maintained and developed by the community on "GitHub".

The class "RfidForm" is derived from the system class "Windows.Forms" and contains the form constructor and the event handlers of the main window of the program interface. This class directly accesses the properties, methods and events of the "UAClientHelperAPI" and the "OPC UA .NET Stack".

All other Forms classes are also derived from the system class "Windows.Forms" and are called via the "RfidForm" button. Depending on the form, they also access the "UAClientHelperAPI" or the "OPC UA .NET Stack".

The "UAGetRfidMethodIdentifiers" class is used to determine the node IDs of the OPC UA methods of an RFID reader. The node IDs are required to call the methods of a server. The OPC UA instances derived from "AutoID" are determined by "browsing".

"UAAutoIDIdentifiers" contains static classes that return the node IDs of the type descriptions of "AutoID". The type descriptions are required for decoding and encoding the specific RFID data types.

"UAAutoIDDDataTypes" contains the actual classes that are transferred for decoding and encoding the specific RFID data types. The individual classes require the node IDs, which are provided via "UAAutoIDIdentifiers".
2.1.2 UAClientHelperAPI

The following explanations describe the freely usable class "UAClientHelperAPI", which implements some basic functions of the OPC UA client.

Class diagram

The UAClientHelperAPI accesses the .NET assemblies "Opc.UA.Client.dll" and "Opc.UA.Core.dll" of the OPC Foundation.

The instantiation of the class additionally returns a "session object", via which you can directly access the "OPC UA .NET Stack".

The following figure shows the class diagram for the class "UAClientHelperAPI". The most important access methods to an OPC UA server are encapsulated and summarized in this class. The diagram shows only the "public" methods, attributes and events.

Figure 2-2
**Method description**

The following table explains the functions of the public methods and -events within the class "UAClientHelperAPI":

Table 2-1

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindServers</td>
<td>Searches for OPC UA servers in the network. Requirements: An LDS (Local Discovery Server) or GDS (Global Discovery Server) must be available.</td>
</tr>
<tr>
<td>GetEndpoints</td>
<td>Determines the available endpoints on a server that can be used to establish a connection.</td>
</tr>
<tr>
<td>Connect</td>
<td>Establishes a connection to a server and creates a secure channel and session to the server.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Ends an existing session and closes the connection to the server.</td>
</tr>
<tr>
<td>BrowseRoot</td>
<td>Returns a collection of nodes found in the root directory of the server.</td>
</tr>
<tr>
<td>BrowseNode</td>
<td>Returns a collection of nodes that can be found hierarchically under a specific node.</td>
</tr>
<tr>
<td>BrowseByReferenceType</td>
<td>Returns a collection of nodes that can be found under the specification of a reference type under a specific node.</td>
</tr>
<tr>
<td>GetNamespaceUri</td>
<td>Returns the namespace Uri to a passed namespace index.</td>
</tr>
<tr>
<td>GetNamespaceIndex</td>
<td>Returns the namespace index to a given namespace Uri.</td>
</tr>
<tr>
<td>GetNamespaceArray</td>
<td>Returns the namespace array.</td>
</tr>
<tr>
<td>Subscribe</td>
<td>Creates a subscription on the server.</td>
</tr>
<tr>
<td>RemoveSubscription</td>
<td>Deletes a specific subscription from the server.</td>
</tr>
<tr>
<td>AddMonitoredItem</td>
<td>Adds a MonitoredItem to an existing subscription for observation.</td>
</tr>
<tr>
<td>AddEventMonitoredItem</td>
<td>Adds an (Event)MonitoredItem to an existing subscription for observation.</td>
</tr>
<tr>
<td>RemoveMonitoredItem</td>
<td>Deletes an existing MonitoredItem of a subscription.</td>
</tr>
<tr>
<td>ReadNode</td>
<td>Reads the metadata of a specific node.</td>
</tr>
<tr>
<td>ReadValues</td>
<td>Reads the values of variable nodes.</td>
</tr>
<tr>
<td>WriteValues</td>
<td>Writes values to variable nodes.</td>
</tr>
<tr>
<td>ReadStructUdt</td>
<td>Reads values of user-defined structures and UDTs using the server's TypeDictionary.</td>
</tr>
<tr>
<td>WriteStructUdt</td>
<td>Writes values to user-defined structures and UDTs that have been previously read.</td>
</tr>
<tr>
<td>RegisterNodeIds</td>
<td>Register node ids with the server for optimized access to them.</td>
</tr>
<tr>
<td>UnregisterNodeIds</td>
<td>Deletes the registration of already registered Node IDs.</td>
</tr>
<tr>
<td>GetMethodArguments</td>
<td>Determines the available input and output arguments of a method.</td>
</tr>
<tr>
<td>CallMethod</td>
<td>Calls a method on the server.</td>
</tr>
<tr>
<td>Notification_CertificateValidation</td>
<td>Event that is fired when a server certificate arrives from the client (when connecting).</td>
</tr>
</tbody>
</table>
Method | Explanation
---|---
Notification_MonitoredItem | Event that is fired when the values of a MonitoredItem have changed.
Notification_KeepAlive | Event that is fired when a KeepAliveNotification arrives.
Notification_MonitoredEventItem | Event that is fired when an OPC UA event arrives.

**Note**
The methods marked in bold are used in the example program.

2.1.3 **OPC UA .NET Stack**

The use of the "OPC UA .NET Stack" is not described in detail in this documentation due to the scope of the SDK. For more information, visit the official section on GitHub for the SDK ([6]) or analyze the open source code of this example.

Basically applies:
The OPC UA client functionality is implemented by the library "Opc.Ua.Client". The client functions require the library "Opc.Ua.Core".

**Opc.Ua.Client**

OPC UA implements client-server communication via so-called "sessions". The class "Session" exists in the library "Opc.Ua.Client". This class contains all functions for communication with a server.

The following classes of the client library are essential for the communication of this client example:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>This class contains all objects and methods for client-server communication and their administration.</td>
</tr>
<tr>
<td>Subscription</td>
<td>This class covers the creation and management of &quot;subscriptions&quot;. A Subscription object is added to the Session object using the Session.AddSubscription() method and activated on the server using the Subscription.Create() method.</td>
</tr>
<tr>
<td>MonitoredItem</td>
<td>This class covers the creation and management of &quot;MonitoredItems&quot;. MonitoredItems specify the nodes of a server whose process values are to be communicated to a client using a subscription. A MonitoredItem object is added to the Subscription object using the &quot;Subscription.AddItem()&quot; method. The activation on the server is done by the method &quot;Subscription.Apply()&quot;. MonitoredItems are mainly used to monitor variable nodes or to receive OPC UA events.</td>
</tr>
</tbody>
</table>

**Opc.Ua.Core**

The library "Opc.Ua.Core" provides all basic classes for the functions of the SKD. In this example, the following classes are particularly important for functionality:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
</table>
| ApplicationConfiguration | The class "ApplicationConfiguration" specifies the OPC UA instance, in this case the OPC UA client. Among other
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>things, this configuration provides settings for security, the certificate, communication restrictions between client and server, and specifies the application name, type, and -URI.</td>
<td></td>
</tr>
<tr>
<td>DiscoveryClient</td>
<td>The class &quot;DiscoveryClient&quot; enables the search for OPC UA servers and asks for the endpoints of the servers. This function is essential if no explicit server information is available to a client before the connection is established.</td>
</tr>
<tr>
<td>Nodeld</td>
<td>The class &quot;Nodeld&quot; is used to address nodes of an OPC UA server. A Nodeld is needed, for example, to read a variable, query node attributes, or call methods.</td>
</tr>
<tr>
<td>Node</td>
<td>The &quot;Node&quot; class is used to describe an OPC UA node. Node contains all attributes of a node. You can use a Nodeld to read a node from the server whose attributes are stored in a node object. The members of Node provide information about the node type and properties.</td>
</tr>
<tr>
<td>EncodableFactory</td>
<td>The &quot;EncodableFactory&quot; class is used for encoding and decoding complex data types (e.g. structures). Use the method &quot;EncodableFactory.GlobalFactory.AddEncodableType()&quot; to add a data type to the class that is to be encoded or decoded. This allows a variable to be converted to the desired data type (cast). The class that should map the complex data type must be derived from &quot;EncodableObject&quot; and have specific &quot;Encodable&quot; or &quot;Decodable&quot; members.</td>
</tr>
</tbody>
</table>
2.2 Operation

The following descriptions and step-by-step instructions show you how to commission and operate this application example.

2.2.1 Commissioning OPC UA servers

Detailed information on commissioning the OPC UA server from RF600 can be found in the bibliography at \[4\]. Information on commissioning the OPC UA server from RF18xC can also be found in the bibliography under \[7\].

2.2.2 Description of the user interface

The structure of the user interface of the example program consists of a menu bar and a task view with two task cards.

Overview

The following illustrations describe the user interface of the example program.

![User Interface Illustration](image-url)

Table 2-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>&quot;File&quot; contains the button &quot;Exit&quot; to exit the program.</td>
</tr>
<tr>
<td>2.</td>
<td>&quot;Server&quot; contains the buttons &quot;Connection&quot; for establishing the connection and &quot;Disconnect&quot; for closing the connection.</td>
</tr>
<tr>
<td>3.</td>
<td>&quot;RFID&quot; contains the OPC UA methods for controlling a reader. This menu is only active if the server supports the &quot;AutoID&quot; specification.</td>
</tr>
<tr>
<td>4.</td>
<td>&quot;Extras&quot; includes the buttons &quot;Clear Logfile&quot; to clear the contents of a log file and &quot;Get Logfile&quot; to create a log file and save the contents of the logger to a file.</td>
</tr>
<tr>
<td>5.</td>
<td>&quot;Help&quot; contains the &quot;About&quot; button, which displays a window with copyright information and the current program version.</td>
</tr>
<tr>
<td>6.</td>
<td>The Task-Card &quot;Logger&quot; contains a text output, which outputs the results of the OPC UA methods and some program information.</td>
</tr>
<tr>
<td>7.</td>
<td>The task card &quot;OPC UA&quot; is the actual work area of the example program.</td>
</tr>
</tbody>
</table>
Work area

Figure 2-4

1. The “Address Space” section shows the address space of the currently connected server.
2. The Node Attributes section displays the attributes of a node selected in the Address Space section.
3. The Subscription Task Card is used to subscribe to variable nodes of the address space. The Task Card “Events” is used to subscribe to nodes with “Event Notifier”.
4. The “Monitored Items” section shows variable nodes that the client subscribes to.
5. The “Settings” button opens a window with configuration settings for the subscription.
6. The “Clear” button empties the subscription and deletes the entries from the “Monitored Items” list.
7. The “Monitored Event Emitter” section shows the subscribed event nodes and the events that arrived.

Table 2-5

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The “Address Space” section shows the address space of the currently connected server.</td>
</tr>
<tr>
<td>2.</td>
<td>The Node Attributes section displays the attributes of a node selected in the Address Space section.</td>
</tr>
<tr>
<td>3.</td>
<td>The Subscription Task Card is used to subscribe to variable nodes of the address space. The Task Card “Events” is used to subscribe to nodes with “Event Notifier”.</td>
</tr>
<tr>
<td>4.</td>
<td>The “Monitored Items” section shows variable nodes that the client subscribes to.</td>
</tr>
<tr>
<td>5.</td>
<td>The “Settings” button opens a window with configuration settings for the subscription.</td>
</tr>
<tr>
<td>6.</td>
<td>The “Clear” button empties the subscription and deletes the entries from the “Monitored Items” list.</td>
</tr>
<tr>
<td>7.</td>
<td>The “Monitored Event Emitter” section shows the subscribed event nodes and the events that arrived.</td>
</tr>
</tbody>
</table>
2.2.3 Putting the OPC UA Client Example into Operation

The sample client is provided through a Visual Studio project.

1. Download the project "109769038_OPC_UA_RFID_CODE_V1_1.zip" to your hard disk. The download can be found on the HTML page of this article (1).

1. Unpack the project.

2. Navigate in the unzipped folder to "RFID_OPC_UA_ReaderApp > RfidOpcUaForm > Compiled". This folder contains the compiled executable EXE file of the OPC UA client example.

3. Start the program by double-clicking on the file "RfidOpcUa.exe".

2.2.4 Establishing a connection to the OPC UA Server

Via the menu item "Connection" you connect the example client with an OPC UA server. Proceed as follows:

1. Click on "Server > Connection" in the menu bar.

2. In the "Server Connection" window, enter the IP address of your OPC UA server in the "Address" text field.

3. Enter the port of your OPC UA server in the text field "Port".

4. Then click "Discover".

5. Click on an available endpoint of your server from the "Discovery" list.

6. (Optional) Activate the "UserName" radio button in the "Authentication Settings" area and enter a user name and password in the corresponding text fields.

7. Click the "Connect" button. The connection status is displayed in the "Connection Status" section.
2.2.5 Browsing the Address Space of the OPC UA Server

In the work area "OPC UA" you browse within the address space of the OPC UA server.

1. Connect the program to an OPC UA server.
2. Switch to the task card "OPC UA".
3. In the tree view "Address Space" you browse through the individual nodes in the address space of the OPC UA server. When you click a node in the tree, the Node Attributes data view on the right provides specific information about the selected node.

2.2.6 Subscribe to variables

You can use a subscription to receive value changes for variables without actively sending read requests for these variables. Proceed as follows to subscribe to a variable:

1. Browse to the desired variable in the "Address Space" tree view.
2. Drag and drop the variable into the "Monitored Items" list. During "publishing", i.e. the transmission of new values from the server, the columns of the list are updated automatically.
3. Edit the "Subscription" settings using the "Settings" button.
4. Edit the settings of the individual "Monitored Items" via the context menu (right click) in the list.

Note
You can only subscribe to variable nodes. You can recognize a variable node by the attribute "Node Class" = "Variable".

2.2.7 Subscribe to events

OPC UA servers according to the Companion specification "AutoID" offer you the function to send OPC UA events. For identification systems, the "scan events" are particularly important here. If you start a "scan" on the RFID device and subscribe to the corresponding event node, the OPC UA client automatically receives scan events if a transponder passes the RF field of the reader. The scan events provide your client with the following information:

- EPCID
- Time stamp
- RSSI value (reception strength)
- Readpoint
- Power (transmitting power of the antenna)
The RF18xC communication modules offer the following additional OPC UA events in addition to "AutoID" with the following information:

- **AutoIdPresenceEvent**
  - Readpoint
  - Presence of a tag

- **AutoIdLastAccessEvent**
  - Readpoint
  - Client that accessed last
  - Last access
  - Result of last access

- **AutoIdLastLogEntryEvent**
  - Readpoint
  - Last log entry

Proceed as follows to subscribe to an event:

1. In the tree view "Address Space" browse to the desired event node.
2. Drag and drop the node into the left list of the "Monitored Event Emitter" section.
3. In the dialog that appears, select the events you want to subscribe to and confirm with “OK”.

![Set Event Filter dialog]

4. In “Publishing”, i.e. the submission of new events from the server, the event data is automatically written to the right list of the “Monitored Event Emitter” section.

![Monitored Event Emitter section]

5. Edit the “Subscription” settings using the “Settings” button.

6. Edit the settings of the individual “Monitored Items” via the context menu (right click) in the left list.

**Note**

You can recognize an event node by the attribute “Event Notifier” = “1”.
2.2.8 Write/read variables

The example client offers you the possibility to read or write OPC UA variables.

Read variables

You read OPC UA variables of a server as follows:
1. Browse to the desired variable in the "Address Space" tree view.
2. Click on the node. The value of the variable is displayed in the "Node Attributes" data view in the "Value" line.

Write variables

You read OPC UA variables of a server as follows:
1. Browse to the desired variable in the "Address Space" tree view.
2. Double-click the variable.
3. Enter the value to be written in the "Write Value" window in the "Write Value" column and confirm with "Apply".

Alternatively, you can use the context menu (right mouse button) of the "Monitored Items" list to open and describe variables already subscribed to in the "Write Value" window. You can select and describe several variables directly.

Note

You can only describe variables with write permission. You can recognize this by the attribute "Access Level" = "3".
2.2.9 Call methods

Via OPC UA methods you control the RFID operations of a reader. The following list explains the existing methods:

Table 2-6

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan</td>
<td>Performs an RFID scan (&quot;inventory&quot;). This method returns the scan result only after a precondition has been reached.</td>
</tr>
<tr>
<td>ScanStart</td>
<td>Starts an RFID scan. The scan is terminated when a precondition is met. With this method, the scan results are transmitted as OPC UA events as long as the scan process continues.</td>
</tr>
<tr>
<td>ScanStop</td>
<td>Stops an RFID scan that was started by &quot;ScanStart&quot;.</td>
</tr>
<tr>
<td>KillTag</td>
<td>Deactivates an RFID transponder</td>
</tr>
<tr>
<td>LockTag</td>
<td>Blocks individual memory areas of an RFID transponder. Locked memory areas can only be accessed with an &quot;Access Password&quot;.</td>
</tr>
<tr>
<td>ReadTag</td>
<td>Reads a desired memory bank of an RFID transponder.</td>
</tr>
<tr>
<td>SetTagPassword</td>
<td>Set the &quot;Access Password&quot; and the &quot;Kill Password&quot; of an RFID transponder.</td>
</tr>
<tr>
<td>WriteTag</td>
<td>Describes a desired storage bank of an RFID transponder.</td>
</tr>
</tbody>
</table>

Call method

Proceed as follows to call a method:

1. In the "Address Space" tree view, browse to one of the methods listed above.
2. Double-click the desired method. Alternatively, you can call up the desired method via the menu item "RFID".

![RFID methods in the Address Space tree view](image)

Note

The methods of an RFID device can only be called if the namespace "AutoID" has been found on the server.
Execute Scan/ScanStart/ScanStop

To start a scan, follow these steps:

1. Call the "Scan" or "ScanStart" method.

2. (Optional) Select the desired reading point in the "Readpoint" drop-down list.

3. Specify a precondition. You have the following choice:
   - "NumberOfCycles": Fixed number of scan cycles
   - "InventoryDuration": Fixed duration of scans
   - "Data available": Scan until a transponder is detected.

4. Click "Call" to execute the method. In the "Output" area you will receive a confirmation of the execution.

5. With the "Scan" method, the scan results are output in the "Output" area. With the "ScanStart" method, the results are transmitted via OPC UA event. For this, please observe chapter 2.2.7 Subscribe to events.

6. To stop a scan started with "ScanStart", call the method "ScanStop". All you must do is select the reading point and execute the method with "Call".
Executing KillTag

Proceed as follows to disable an RFID transponder:

1. Call the "KillTag" method.
2. Enter the EPCID of the transponder in the "Tag Identifier" field. For example, you can determine the EPCID using a scan.
3. Enter the kill password of the transponder in the "Password" field. For more information on the Kill password, see the "Run SetTagPassword" section of this chapter.
4. Click "Call" to execute the method. In the "Output" area you will receive a confirmation of the execution.

Note

Deactivating a transponder in permanent.
Executing LockTag

1. Call the "LockTag" method.
2. Enter the EPCID of the transponder in the "Tag Identifier" field. For example, you can determine the EPCID using a scan.
3. Enter the access password of the transponder in the "Password" field. For more information on the Access password, see the "Run SetTagPassword" section of this chapter.
4. Select the area to be locked or unlocked in the "Region" drop-down list.
5. In the "Lock" drop-down list, select whether you want to lock or unlock the area.
6. Click "Call" to execute the method. In the "Output" area you will receive a confirmation of the execution.

Note
An EPCID is only required if there are several transponders in the RF field.
Executing ReadTag

1. Call the "ReadTag" method.
2. Enter the EPCID of the transponder in the "Tag Identifier" field. For example, you can determine the EPCID using a scan.
3. (Optional) Enter the access password of the transponder in the "Password" field. For more information on the Access password, see the "Run SetTagPassword" section of this chapter.
4. Select the memory bank of the transponder to be read out in the "Region" drop-down list. Further information on the structure of the memory banks can be found in the manual "SIMATIC Ident RFID Systems SIMATIC RF600" (4).
5. Assign a memory address in the text field "Offset" from which the area is to be read out.
6. Specify the length of the data to be read in the "Length" text box. The length is specified with decimal values and describes how many bytes are to be read.
7. Click "Call" to execute the method. In the "Output" area you will receive a confirmation of the execution and the read result.

Note
An EPCID is only required if there are several transponders in the RF field.
Execute SetTagPassword

1. Call the "SetTagPassword" method.
2. Enter the EPCID of the transponder in the "Tag Identifier" field. For example, you can determine the EPCID using a scan.
3. (Optional) Enter the access password of the transponder in the "AccessPassword" field.
4. Select the transponder password to be changed in the "PasswordType" drop-down list.
5. Enter a new password in the "NewPassword" text box.
6. Click "Call" to execute the method. In the "Output" area you will receive a confirmation of the execution.

Note

An EPCID is only required if there are several transponders in the RF field.
Executing WriteTag

1. Call the "WriteTag" method.
2. Enter the EPCID of the transponder in the "Tag Identifier" field. For example, you can determine the EPCID using a scan.
3. (Optional) Enter the access password of the transponder in the "Password" field.
4. Assign a memory address in the text field "Offset" from which the area is to be written.
5. Enter the data to be written in the "Data" text field.
6. Click "Call" to execute the method. In the "Output" area you will receive a confirmation of the execution.

Note: An EPCID is only required if there are several transponders in the RF field.
3 Useful information

3.1 Certificate handling in the example program

The client certificate of the sample program is created the first time it is started and stored in the Windows certificate store. The certificate is required for encrypted and signed connections and bears the name "UA Client RFID". The certificate is self-signed by the application.

The certificate store can be found under "Start > Run > mmc > Enter > File > Add/Remove Snap-In > Certificates > Add > Select My Account > OK" in the folder "My Certificates".

To recreate the certificate, delete it from the store and then restart the application.

Note: The sample client automatically accepts all server certificates unchecked for each session, but does not save them.

3.2 Information about the license model of the software

The "OPC UA .NET Stack" used in this example was created by the OPC Foundation and will be continued as a joint project. The GitHub community is responsible for the further development and maintenance of the stack.

Depending on your membership in the OPC Foundation, the stack is subject to two different license models:

1. The "RCL" model applies to members of the OPC Foundation.
2. For non-members of the OPC Foundation the model "GPL 2.0" applies.

Note: Before you start developing your own application, check out the appropriate licensing models based on your membership of the OPC Foundation.

Note: The crypto component "BouncyCastle.Crypto.dll" and the implementations of Siemens AG are subject to the open source license "MIT".
4 Appendix

4.1 Siemens services

Industry Online Support

Do you have any questions or need assistance?
Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.
The Industry Online Support is the central address for information about our products, solutions and services.
Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:
https://support.industry.siemens.com

Technical Support

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

Service offer

Our range of services includes the following:
• Product trainings
• Plant data services
• Spare parts services
• Repair services
• On-site and maintenance services
• Retrofitting and modernization services
• Service programs and contracts
You can find detailed information on our range of services in the service catalog web page:
https://support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for Apple iOS and Android:
https://support.industry.siemens.com/cs/ww/en/sc/2067
4.2 Links and literature

Table 4-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>This article in Siemens Industry Online Support <a href="https://support.industry.siemens.com/cs/ww/de/view/109769038">https://support.industry.siemens.com/cs/ww/de/view/109769038</a></td>
</tr>
<tr>
<td>5</td>
<td>Resources of the OPC Foundation <a href="https://opcfoundation.org/">https://opcfoundation.org/</a></td>
</tr>
</tbody>
</table>

4.3 Change documentation

Table 4-2

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>08/2019</td>
<td>First version</td>
</tr>
<tr>
<td>V1.1</td>
<td>03/2020</td>
<td>Extension about SIMATIC-Events, error corrections</td>
</tr>
</tbody>
</table>