SIEMENS

SIMATIC Ident

RFID systems
OPC UA for SIMATIC Ident

Application Manual

02/2022
C79000-G8976-C661-01

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.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

C79000-G8976-C661-01
Ⓟ 02/2022 Subject to change

Table of contents

Introduction

1

Purpose of this documentation

This manual contains information required to program SIMATIC Ident devices via the OPC UA interface.

Basic knowledge required

Specific knowledge of OPC UA programming, as well as general knowledge of automation technology and identification systems, is required to understand this manual.

Trademarks

The following and possibly other names not identified by the registered trademark sign ® are registered trademarks of Siemens AG:

SIMATIC ®, SIMATIC RF ®, MOBY ®, RF MANAGER ®, SIMATIC Sensors ®

Orientation in the documentation

In addition to this manual, you may need the hardware manuals of the particular Ident devices that you want to program. At this time, the following SIMATIC Ident products can be programmed via the OPC UA interface:

- SIMATIC RF360R
- SIMATIC RF610R, RF615R, RF650R, RF680R and RF685R
- SIMATIC RF185C, RF186C, RF188C, RF186CI and RF188CI
- SIMATIC RF166C

You can find the current versions of the relevant manuals on the pages of the Siemens Industry Online Support (https://support.industry.siemens.com/cs/ww/en/ps/14970/man).

Validity

This document is valid for all products listed here. This products are referred to generally as "Ident devices" hereinafter.

Abbreviations and naming conventions

The following terms/abbreviations are used synonymously in this document:

Transponder, tag	Data medium, mobile data storage (MDS)
Communications module (CM)	Interface module (ASM)

1.1 Security information

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

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

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit
https://www.siemens.com/industrialsecurity.

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

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

Application description

2

OPC UA is a standardized communication protocol. It allows data exchange between all types of industrial devices that support OPC UA and that are integrated in the same network. In this context, devices that provide or publish data, information and command calls are known as OPC UA servers. Devices that use this data, information and these command calls are known as OPC UA clients.

Using OPC UA, you can also integrate the SIMATIC Ident devices into cloud systems or communicate with them.

Knowledge of fundamental OPC mechanisms as well as programming know-how are essential in order to understand the following section and to implement your own OPC UA client to use with the Ident devices. The OPC UA standard specifications will help you here.

The standard "OPC Unified Architecture for AutoID Devices" was defined by the organizations "AIM Germany" and "OPC Foundation". This describes the connection of identification devices via OPC UA. The identification devices can be subdivided as follows:

- Text recognition devices (OCR)
- Optical readers (e.g. barcode)
- RFID reader and
- Devices for localization (RTLS).

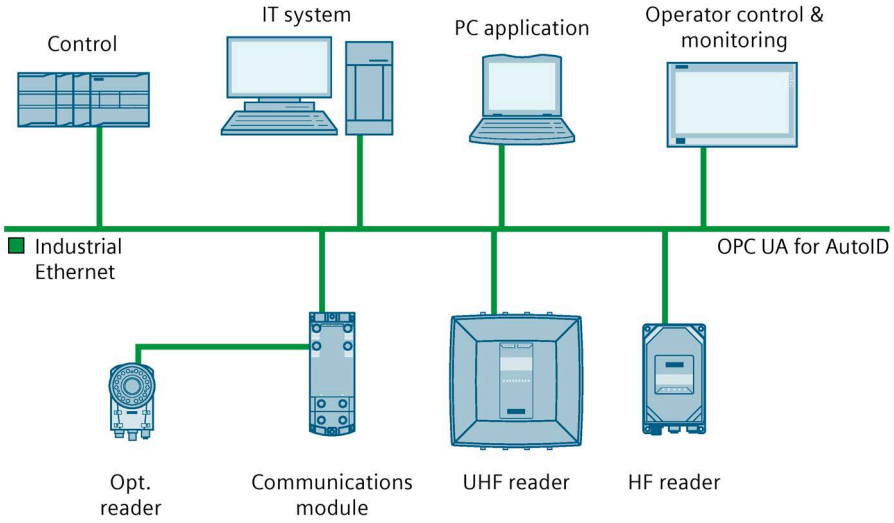

Figure 2-1 Ident devices in an OPC UA network

You will find more information on OPC UA on the pages of the "OPC Foundation (https://opcfoundation.org/)". The "OPC Unified Architecture for AutoID Devices Specification" companion specification can be obtained via the "AIM Germany (www.aim-d.de)" or the "OPC Foundation".

2.1 Operating principle

At this time, the following SIMATIC Ident products can be programmed via the OPC UA interface:

- SIMATIC RF360R
- SIMATIC RF610R, RF615R, RF650R, RF680R and RF685R
- SIMATIC RF185C, RF186C, RF188C, RF186CI and RF188CI
- SIMATIC RF166C

The OPC UA-capable SIMATIC Ident devices have implemented OPC UA servers with the range of functions for RFID devices defined by "OPC Unified Architecture for AutoID" (Release 1.01). For this, "OPC Unified Architecture for AutoID Devices" defines the "AutoIdDevice" and, derived from this, the "RfidReaderDevice". The "AutoIdDevice" and "RfidReaderDevice" are OPC UA definitions for read points.

Address areas and read points

Each OPC UA server publishes to the OPC UA clients its OPC UA functionalities via nodes in its address area. Like all OPC UA objects, the "RfidReaderDevice" is an OPC UA node with defined attributes such as "BrowseName" and "DisplayName". You can find the read points or "RfidReaderDevices" under the "DeviceSet" node in the address area of the OPC UA servers of the SIMATIC Ident devices.

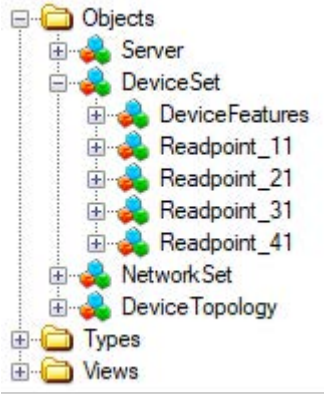

Figure 2-2 "DeviceSet" node

The Ident devices support a different number of read points. Each read point stands for an independent "AutoIdDevice" or "RfidReaderDevice" that can be addressed independently of one another.

Device-specific structure

SIMATIC RF18xC/RF18xCI and RF166C

With the RF18xC/RF18xCI and RF166C communications modules, each reader interface stands for a read point and thus for an independent "AutoIdDevice" or "RfidReaderDevice". Consequently, the number of displayed read points depends on the number of devices connected to the communications module.

Table 2- 1 Number of read points

Ident device	Number of read points
RF185C	1
RF186C	2
RF188C	4
RF186CI	2
RF188CI	4
RF166C	2

The assignment of the OPC UA read points to the connected devices is derived from the read point names.

Table 2- 2 Assignment of the OPC UA read points (based on an RF188C)

Reader interface	BrowseName	DisplayName [1)]
1	Read_point_1	Readpoint_11
2	Read_point_2	Readpoint_21
3	Read_point_3	Readpoint_31
4	Read_point_4	Readpoint_41

1) The name can be changed/adapted.

SIMATIC RF360R

The RF360R reader has a read point and thus an independent "AutoIdDevice" or "RfidReaderDevice".

Table 2- 3 Read points and assignment

Ident device	Number of read points	BrowseName	DisplayName [1)]
RF360R	1	Read_point_1	Readpoint_11

1) The name can be changed/adapted.

SIMATIC RF600

With the RF600 readers, the number of read points and thus the independent "AutoIdDevices" or "RfidReaderDevices" depend on the configuration. As soon as read points have been assigned to antennas, the relevant read point is displayed.

Table 2- 4 Number of read points

Ident device	Number of read points
RF610R	1
RF615R	1 ... 2
RF650R	1 ... 4
RF680R	1 ... 4
RF685R	1 ... 2

The assignment of the OPC UA read points to the connected and configured devices is derived from the read point names.

Table 2- 5 Assignment of the OPC UA read points (based on an RF650R or RF680R)

Read point	BrowseName	DisplayName [1)]
1	Read_point_1	Readpoint_1
2	Read_point_2	Readpoint_2
3	Read_point_3	Readpoint_3
4	Read_point_4	Readpoint_4

[1)] The name can be changed/adapted.

2.2 OPC UA basic functions

OPC UA basic functions

The integrated OPC UA servers of the Ident devices support the following OPC UA basic functions:

- OPC UA server basic functions according to the "Embedded 2017 UA Server Profile" of the OPC Foundation.

  As an extension of the "Embedded 2017 UA Server Profile":

  – "Standard Event Subscription Server Facet"

  – "SecurityPolicy - Basic128Rsa15"

  – "SecurityPolicy - Basic256"

  – "SecurityPolicy - Aes256-Sha256-RsaPss"

  – Maximum five OPC UA client connections
- "Full AutoID Server Facet" according to the specification "OPC Unified Architecture for AutoID" (Release 1.01)
- "Device Integration Model (DI)" according to the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03).

  This is the basis specification for field devices and defines the object type "DeviceType" from which the "AutoIdDevice" or "RfidDevice" object is derived in turn.
- "Global Certificate Management Server Facet" (GDS push)
- "SoftwareUpdateType" for the firmware update

Operating mode

An OPC UA client can be connected to an Ident device as a main application or as a second application connected in parallel to a PROFINET, PROFIBUS or EtherNet/IP main application for diagnostics purposes.

For an OPC UA client to also be able to connect to an existing application, the "Parallel" parameter must be selected in the Web-based management (WBM) of the device in the "Communication > OPC UA" menu. Because the OPC UA client only has diagnostic access to the Ident device in this scenario, write access is no longer possible for OPC UA clients.

Encryption

OPC UA provides the possibility of encrypted communication and application authentication as well as user authentication. You are urgently recommended to make use of these possibilities in order to protect the OPC UA server itself and the communication between OPC UA server and OPC UA clients from access from third parties. The specified protective mechanisms can be configured in the WBM of the device in the "Communication > OPC UA" menu.

2.3 OPC UA basic mechanisms

The "RfidReaderDevice" from the "OPC Unified Architecture for AutoID Devices" specification and thus the integrated OPC UA servers of the SIMATIC Ident devices offer OPC UA clients three basic mechanisms to query information, make settings and run RFID commands:

- Variables
- Events
- Methods

2.3.1 Variables

OPC UA variables and properties represent a simple way to query information from the Ident device or make settings on the reader. The information can be queried by one-off reading of the variable value or by logging on to the variables and then reading out on value change.

Variables are contained directly in the address area of the servers underneath a read point or an "RfidReaderDevice".

Figure 2-3 Examples of "Variables" below a read point node

Almost all OPC UA clients support the "DataAccess" OPC UA service and therefore variables. However, you need to observe the following points when using variables.

The restriction of the update rate by the "Sampling" interval and by the "Publishing" interval is common to all OPC UA variables. These are fundamental OPC mechanisms that can define the intervals at which the values can be updated or queried via OPC UA. If the intervals at which the values for a variable are updated from a process should be shorter than the specified intervals, values can be overwritten before the query by an OPC UA client.

This problem is worsened by the use of logically related variables. If it is not possible to determine in time that related variables can be queried completely by the client before variables are written with new values again from the process, usage is not possible. If a client supports events, make sure that these are used. The effects described above cannot occur with events.

If the configuration limits are too large due to a small publishing/sampling interval and a large number of variables, it is not possible to adhere to the selected intervals. In this case, increase the interval or reduce the number of variables.

2.3.2 Events

OPC UA events represent a secure way to query or accept information from the Ident device. To receive certain events, the client needs to log on to an event-generating OPC UA node for this event type. After logon, every event generated by this OPC UA node that corresponds to the selected event type is sent to the client by the server. This takes place until the client logs off from these events again.

Events are not contained directly as nodes in the address area of the servers underneath a read point or an "RfidReaderDevice". Instead, the read point only contains references to the event types that can generate them. A generic client can use this, for example, to show a logon screen to the user.

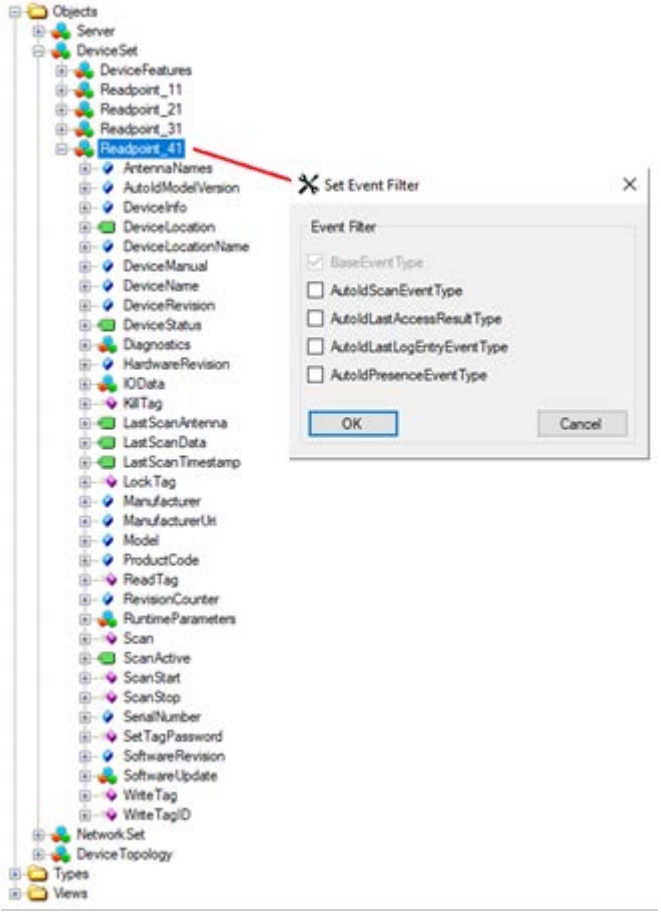

Figure 2-4 "Event" call options of a read point node

The known difficulties in time-based synchronization between the query and update of the variable values during more complex processes with OPC UA variables do not occur with events. If events are defined for specific actions, an independent event is generated for each action that has occurred and been completed. Overwriting of the related information is excluded.

2.3.3 Methods

OPC UA methods represent another secure way to query or accept information from the Ident device or to make settings or run actions on the Ident device. This is done by calling the method by the client. Input parameters that control the sequence are transmitted with most methods. The method is run by the OPC UA server. After this is finished, the results or information are returned to the client via return values and output parameters of the method.

Methods are contained directly in the address area of the servers underneath a read point or a "RfidReaderDevice".

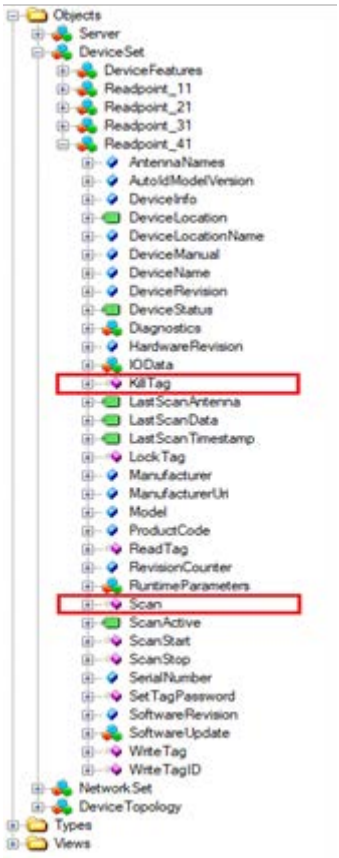

Figure 2-5 Examples of "Methods" below a read point node

The known difficulties in time-based synchronization between the query and update of the variable values during more complex processes with OPC UA variables do not occur with methods. The return values and output parameters returned with a method call are unique and consistent. They are regenerated for each method call.

2.4 Programming

2.4.1 Prerequisites and support

Prerequisites, requirements and limitations

When working with the Ident devices via the OPC UA interface, note the following prerequisites, requirements and limitations:

- A maximum of five OPC UA client connections are permitted. Note that multiple clients can also work with one read point.
- An OPC UA client that wishes to use the full RFID functionality of the Ident device needs to support the following fundamental OPC UA access mechanisms:
  - Data Access (DA)
  - Events
  - Methods
- If possible, always use encrypted endpoints to protect communication between the OPC UA server and the OPC UA client from access by third parties.
- Make sure that no unauthorized persons have access to the communications module settings. It is recommended to leave authentication enabled in the WBM of the Ident device and to create new user profiles.
- The "BrowseName" of the nodes should preferably be used to address OPC UA nodes. These are usually specified by OPC UA standards, whereas "NodeIds" and "DisplayName" can vary depending on the device/manufacturer.

Support in work

There is a large number of tools (files, demo applications) that facilitate working with the Ident devices via the OPC UA interface:

- The "OPC Unified Architecture for AutoID Devices Specification" companion specification can be obtained via the "AIM Germany (www.aim-d.de)" or the "OPC Foundation". The associated "Information Model" file and "Nodeset" file
  - Opc.Ua.AutoID.NodeSet2.xml
  - Opc.Ua.AutoID.NodeIds.csv

  are linked in the "A.1 Namespace and identifiers for AutoID Information Model" appendix of the Companion Specification and can be downloaded via these links.
- An "Information Model" file and "Nodeset" file of the "SimaticIdent.RFxxxx" of the SIMATIC Ident devices derived from "OPC Unified Architecture for AutoID Devices"
  - SimaticIdent.RFxxxx.xml
  - SimaticIdent.RFxxxx.csv

  are stored on the Ident devices and can be downloaded via the WBM of the respective device ("System > Device settings > Device description file").
- The specification "OPC Unified Architecture Part 100: Devices" (Release 1.03) can be obtained from the "OPC Foundation".
- As a basis for the implementation of OPC UA clients, there is a large number of commercial or freely available OPC UA stacks or SDKs for different operating systems and programming languages. This greatly simplifies the development of OPC UA client applications.
- Siemens offers a Windows.NET-based OPC UA demo application of an OPC UA client specifically for Ident devices. This application serves as a sample for testing the RFID functionalities of the Ident devices and as a basis for creating your own OPC UA client.

  You can find the demo application including source code files and documentation as download on the pages of OPC UA client example for RFID devices (https://support.industry.siemens.com/cs/en/de/view/109769038).

Note

Disclaimer of liability

Note that Siemens AG accepts no liability for the demo application "OPC UA .NET Client for SIMATIC RFID devices".

2.4.2 Supported functions

Parallel to regular operation via PC or SIMATIC controller, you can read out the results of the SIMATIC commands and diagnostic information via the Ethernet interface using OPC UA and make them available to another application. Note that only read-only access is supported during parallel operation of Ident devices via OPC UA. In this case, all control or write access operations, as well as calls of methods and setting of variables described in the following, are not supported.

RFID-specific functions

The integrated OPC UA servers of the SIMATIC Ident devices support the following RFID-specific functions based on the specifications "OPC Unified Architecture for AutoID" (Release 1.01), as well as general field device functions according to the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03).

You can find the functions listed below under the following path in the address area of the read point "Objects > DeviceSet > Readpoint_x1" or "Objects > DeviceSet > Read_point_x".

Table 2- 6 RFID-specific variables

Variables	Description
AntennaNames	Antennas of read point
AutoIdModelVersion	Version of the supported AutoID model
DeviceInfo	Information about the device These values are obtained from the WBM start page as soon as the device is restarted. Changes are not transferred to the WBM.
DeviceLocation	Information about the device location These values are obtained from the WBM start page as soon as the device is restarted. Changes are not transferred to the WBM.
DeviceLocationName	Information about the device location These values are obtained from the WBM start page as soon as the device is restarted. Changes are not transferred to the WBM.
DeviceManual	URL to the manual for the device
DeviceName	"DisplayName" of the read point
DeviceRevision	Reserved
DeviceStatus	Device status of read point • 0: Idle The read point is in the "Idle" state. • 1: Error An antenna fault has occurred at the read point. • 2: Scanning The read point is scanning for transponders. • 3: Busy The read point is performing a transponder command.
Diagnostics	Diagnostics information Note: The prerequisite is that this functionality was enabled in the WBM in the "Communication > OPC UA" menu.

Variables		Description
Diagnostics – Presence		Shows the presence of transponders. The prerequisite is that the Ident device supports "Presence" mode. • 0: There is no transponder in the antenna field. • 1: There is at least one transponder in the antenna field. • >1: Exact number of transponders located in the antenna field
Diagnostics – LastAccess		Information about the last successful transponder access or command The possible values of the "LastAccess" variable are not listed here. You can find them in the section ""LastAccess" variables/event (Page 39)".
	Client	Client interface via which the last transponder access on this read point took place.
	Command	Command that was executed during the last transponder access operation.
	Identifier	EPC ID/UID of the transponder which was accessed with the last command.
	Timestamp	Time of the last transponder access
	RWData	Read/written data of the last command. Only possible with commands for reading/writing the transponder memory.
	Antenna	Antenna via which the transponder was accessed with the last command.
	CurrentPowerLevel	Power with which the last command was executed on the transponder. This function is only supported by UHF readers.
	PC	Protocol Control Word of the transponder that was accessed last. This function is only supported by UHF readers.
	Polarization	Polarization with which the last command was executed on the transponder. This function is only supported by UHF readers.
	Strength	RSSI value with which the last command was executed on the transponder. This function is only supported by UHF readers.
Diagnostics – Logbook		Various log information
	LastLogEntry	The last entry in the log as formatted string Example: 2018-05-30T13:14:43.546+00:00 | COMMANDS | readTagIDs | CMD | 0 | Read-point_11 | 000000005575B67D | PC=0000 | RSSI=0 | Pwr=0
	LogColumns	The headers column of the log as formatted string Example: Date_Time ; Type ; Entry ; Command_Type; Sequence_Number ; Readpoint ; EPCID_UID ; PC ; RSSI ; Power
HardwareRevision		Hardware version of the device
LastScanData		Most recently scanned transponder of the read point The scan process can have been triggered via the "Scan", "ScanStart" methods, the "ScanActive" variable or the read point itself (e.g. trigger, IOs). Note: If the "RSSI events" parameter was enabled in the WBM in the "Communication > OPC UA" menu, the filtered out transponders are also reported. Note that this variable is defined as "BaseDataType". The actual data type is defined during runtime by the "CodeTypes" variable. The types "String", "ByteString" and "ScanDataEpc" are supported. Example (RF300 reader, data type string): 000000005575B67D

Variables	Description
LastScanAntenna	Antenna with which the most recently scanned transponder of the read point was detected. The scan process can have been triggered via the "Scan", "ScanStart" methods, the "ScanActive" variable or the read point itself (e.g. trigger, IOs).
LastScanRSSI	RSSI value with which the most recently scanned transponder of the read point was detected. The scan process can have been triggered via the "Scan", "ScanStart" methods, the "ScanActive" variable or the read point itself (e.g. trigger, IOs). This function is only supported by UHF readers.
LastScanTimestamp	Time at which the most recently scanned transponder of the read point was detected. The scan process can have been triggered via the "Scan", "ScanStart" methods, the "ScanActive" variable or the read point itself (e.g. trigger, IOs).
Manufacturer	Manufacturer (always "Siemens AG")
ManufacturerUri	Unique domain name of the device manufacturer (always "https://www.siemens.com")
Model	Variant of the device
ProductCode	Article number of the device
RevisionCounter	Always "-1"
RuntimeParameters	Device settings
RuntimeParameters – CodeTypes	Display of the data type for all "AutoID identifiers" in "AutoID Standard". The setting has an effect specifically on the data type of the "LastScanData" variable, the "Identifier" diagnostics variable and the "ScanData" union used for the "Identifier" in methods or events. The types "String", "ByteString" and "ScanDataEpc" are supported. Possible values: • 0: ByteString • 1: String • 2: ScanDataEpc The data type of this variable is a UInt32 array. Because it is not possible for multiple data types to be defined at the same time, the array consists of exactly one element.
RuntimeParameters – CodeTypesRWData	Definition of the data type for the "RWData" diagnostics variable and the "ScanData" union used for "RWData" in the "RfidAccessEvent". The types "String", "ByteString" and "ScanDataEpc" are supported. Possible values: • 0: ByteString • 1: String • 2: ScanDataEpc The data type of this variable is a UInt32 array. Because it is not possible for multiple data types to be defined at the same time, the array consists of exactly one element.

Variables		Description
RuntimeParameters – RfidSettings – Antenna_xx		RFID-specific antenna settings
	MinRssi	Lowest accepted RSSI value of the antenna This function is only supported by UHF readers. Transponders which are detected with a lower RSSI value are counted as not detected. This is a value without a unit and without direct reference to the power strength. Possible values: • 0 ... 255
	RfPower	Radiated power of the antenna Possible range of values: 0 ... 127 The special features of the different reader types are described below: **With RF380R:** • 2: 0.5 W • 3: 0.75 W • 4: 1.0 W • 5: 1.25 W • 6: 1.5 W • 7: 1.75 W • 8: 2.0 W Settings outside the specified values mean that the default value of "1.25 W" is used. **With UHF readers:** Radiated power of the antenna in [dB] • 0 • 5 ... 33 Settings outside of the specified values have the effect that the respective area limits are set (1...4 ≙ 5; 34...127 ≙ 33).
RuntimeParameters – ScanSettings		Specific scan settings of the "ScanActive" variable Via these settings, you can define how long a scan process started via the "ScanActive" variable takes. The first criterion that is met ends the scan process.
	CodeType	Format of the "LastScanData" variable as string. The types "String", "ByteString" and "ScanDataEpc" are supported. Possible values: • RAW:BYTES • RAW:STRING • EPC
	Cycles	Duration of the scan process based on performed scan cycles. 0 ≙ infinity
	DateAvailaible	Duration of the scan process Possible values: • True: The scan process is ended when the first transponder is detected. • False: The scan process is ended by the stop criteria "Cycles" and "Duration".
	Duration	Duration of the scan process in [ms] 0 ≙ infinity

Variables	Description
ScanActive	Outputs whether the read point is currently scanning for transponders or allows the scan process to be activated/deactivated. This Boolean variable offers simple clients restricted to "DataAccess" the possibility to start scanning the read point. The scan results can be read out via the "LastScan..." variables (e.g. LastScanData). Query of the current status: • TRUE: Read point scans for transponders. • FALSE: Read point does not scan for transponders. Set: • TRUE: The scan process of the read point for transponders is activated. • FALSE: The scan process of the read point for transponders is deactivated. In addition, the scan process is ended when a criterion of the associated "ScanSettings" variable is met.
SerialNumber	Serial no. of the device
SoftwareRevision	Software version of the device

Table 2- 7 RFID-specific events

Events		Description
RfidScanEventType		"TagEvents" of the read point
AutoIdDiagnosisEvent		Diagnostics events of the read point
	RfidLastAccessEvent	Event for the last successful transponder access
	AutoIdLastLogEntryEvent	Event for the last log entry
	AutoIdPresenceEvent	Event for the presence of transponders

Table 2- 8 RFID-specific methods

Methods	Description
Scan	Synchronous execution of inventories The detected transponders are directly returned.
ScanStart, ScanStop	Trigger the read points to start inventories. The detected transponders are delivered by means of events (RfidScanEventType).
KillTag	Destroy transponders. This function is only supported by UHF readers.
LockTag	Lock areas on the transponder. This function is only supported by UHF readers.
SetTagPassword	Set transponder-specific passwords. This function is only supported by UHF readers.
ReadTag	Read out transponder data.
WriteTag	Write transponder data.
WriteTagID	Write an EPC ID/UID including the Protocol Control Word. This function is only supported by UHF readers.

You can find a more exact description of these functions in the specifications "OPC Unified Architecture for AutoID" (Release 1.01), "OPC Unified Architecture Part 100: Devices" (Release 1.03) and the section "Application examples (Page 27)".

Additional RFID-specific functions

The integrated OPC UA server of the SIMATIC Ident devices offer additional functions created by Siemens as a supplement to the AutoID standard. The following table provides an overview and description of the additional functions.

You can refer to the device description files supplied with each Ident device for the IDs of the nodes described here. Alternatively, you can also find the node IDs by browsing through the address area of the server using a generic OPC UA client.

You can find the functions listed below under the following path in the address area of the read point "Objects > DeviceSet > Readpoint_x1" or "Objects > DeviceSet > Read_point_x".

Table 2- 9 Additional RFID-specific functions

Function		Description
RuntimeParameters		Device settings
	SimaticIdentMod-elVersio	Version information of the supported SIMATIC Ident OPC UA model
	DeviceClock	Internal device clock of the device Note: The SIMATIC RF18xC/RF18xCI communications modules and the SIMATIC RF360R reader lose the time when switched off. The time can be reset after switch-on via the application using this variable. Example: 2018-05-29T08:29:15.812Z
RuntimeParameters – CommonSettings		General device settings
	CodeTypes	The parameter is available solely for compatibility reasons and should no longer be used. The function is identical to the parameter "RuntimeParameters > CodeTypes". except that the data type is a single UInt32.
RuntimeParameters – Configuration		Configuration information and backup/restore of the configuration
	ConfigData	The configuration saved in the device This variable serves to read/write the saved configuration and thus to implement the back-up/restore mechanism. Output format: Alphanumeric text of the complete configuration in XML format.
	ConfigID	Unique identifier of the configuration saved in the device If the value of this variable changes, this indicates that the saved configuration of the device has been changed (e.g. via the WBM). This variable serves to monitor the saved configuration and thus to implement the backup/restore mechanism. Output format: Alphanumeric text from hexadecimal characters. Example: 5FAAC0DE
	Configuration	Backup/restore of the configuration via the OPC UA SoftwareUpdate model from the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03). Caution: This function cannot be used at the current time!

Function	Description
IOData – DigitalIOPorts	Information about the digital inputs/outputs
DigitalInputs	States of the digital inputs of the device This function is only supported by Ident devices with digital inputs/outputs. With the RF18xCI communications modules, the number of actually existing physical digital inputs depends on the communications module version being used, the operating mode of the digital inputs/outputs and on the connected IO-Link module. Possible values: 00000000 ... 11111111 Binary characters (0, 1) per input Each position stands for a digital input of the communications module: • Inport00: 1st position (least significant bit right) • ... • Inport07: 8th position Depending on the value of the particular position, the corresponding input is at "ON" (1) or "OFF" (0).
DigitalOutputs	States of the digital outputs of the device This function is only supported by Ident devices with digital inputs/outputs. With the RF18xCI communications modules, the number of actually existing physical digital outputs depends on the communications module version being used, the operating mode of the digital inputs/outputs and on the connected IO-Link module. Possible values: 00000000 ... 11111111 Binary characters (0, 1, x) per output Each position stands for a digital output of the communications module: • Outport00: 1st position (least significant bit right) • ... • Outport07: 8th position Read: Depending on the value of the particular position, the corresponding output is at "ON" (1) or "OFF" (0). Write: Depending on the value of the particular position, the corresponding output is set to "ON" (1) or "OFF" (0). Outputs whose state is to remain unchanged can be masked out with "x". Example: xxxxxx01 ≙ Outport01 is switched off, Outport00 is switched on, the other outputs remain unchanged.
SoftwareUpdate	Firmware information and firmware update of the OPC UA SoftwareUpdate model from the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03).
SoftwareUpdate – Loading	Transferring new firmware to the device Note: The firmware has not yet been activated.
SoftwareUpdate – Loading – CurrentVersion	Information about the firmware currently active on the device
Manufacturer	Firmware publisher (always "Siemens AG")
ManufacturerUri	Unique domain name of the device manufacturer (always "https://www.siemens.com")
SoftwareRevision	Version of the firmware
ErrorMessages	Error description of the errors that may have occurred during loading of the firmware.
SoftwareUpdate – Loading – FileTransfer	Actual loading of the new firmware Object of the type "TemporaryFileTransfer". Through this object, a temporary "FileType" object is generated which makes the file methods available for actual transfer.

Function	Description
ClientProcessingTimeout	Maximum wait time that the server accepts when transferring between the individual firmware blocks. If this time is exceeded by the client, the server considers the transfer to have failed. Always 20 000 ms or 20 s.
CloseAndCommit	Method If the firmware was completely transferred by a call of "GenerateFileForWrite" and subsequent calls of the "Write" method of the temporarily created "FileType" object, the transfer can be ended by this method. The firmware is now completely available as "PendingVersion" on the Ident device. The temporarily created "FileType" object is destroyed again. Note: If a "Rebrowse" of the address area is performed subsequently in a generic client, the "TemporaryFile" node attached temporarily under "SoftwareUpdate – Loading – FileTransfer", which represents the "FileType" object, is removed again.
GenerateFileForRead	Method Caution: This method is not supported. It is not possibly to read back previously transferred firmware from the device.
GenerateFileForWrite	Method Generation of a temporary "FileType" object to write or transfer the new firmware to the device via file operations. The value 1 ("Pending") must always be transmitted as values for the "GenerateOptions" input parameter. Other values are not accepted. The "FileNodeId = 100000" of the generated temporary "FileType" object is returned as output parameter. This must be used to address the temporary "FileType" object to be able to call the methods for the file operations. In addition, a "FileHandle" is returned as output parameter. This must be specified as input parameter for all file operations via the temporary "FileType" object. Note: If a "Rebrowse" of the address area is performed subsequently in a generic client, there is now a "TemporaryFile" node under "SoftwareUpdate – Loading – FileTransfer". This represents the temporary "FileType" object. The firmware can then be transferred to the device block-by-block by repeated calls of the "Write" method of the temporary "FileType".
GetUpdateBehaviour	Method Query of the update behavior of a firmware version specified via input parameter. With the Ident devices, this always relates to transferred firmware that is not yet activated. The "ManufacturerUri" and "SoftwareRevision" of the "Pending" firmware must be transmitted as input parameters. The return value is always "11" (decimal) or "1011" with binary encoding. Meaning of the bits: • Bit0: Keeps parameters The selected parameters are retained. • Bit1: OPC UA Server will restart during installation The OPC UA server is restarted during the installation. • Bit3: Device will reboot during update The device is restarted during the update. You can find detailed information on this in "UpdateBehavior OptionSet" of the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03).
SoftwareUpdate – Loading – PendingVersion	Information about firmware transferred to the device but not yet activated
Manufacturer	Firmware publisher (always "Siemens AG")
ManufacturerUri	Unique domain name of the device manufacturer (always "https://www.siemens.com")
SoftwareRevision	Version of the firmware

Function			Description
		WriteBlockSize	Maximum size of a write block Always 1 048 576 bytes or 1 024 KB. The firmware of the Ident devices must be transferred in blocks with this maximum size.
	SoftwareUpdate – Installation		Activation of firmware already transferred to the Ident device but not yet activated. Note: Immediately after successful activation, the Ident device performs a restart independently. The OPC UA connection is lost. It must be restored after subsequent startup of the device
		CurrentState	Shows the current update state. • 1: Idle • 2: Installing • 3: Error during installing
		InstallSoftware-Package	Method Activation of the firmware previously transferred to the device ("Pending" firmware). The "ManufacturerUri" and "SoftwareRevision" of the "Pending" firmware must be transmitted as input parameters. Note: Immediately after successful activation, the Ident device performs a restart independently. Note that the OPC UA connection is lost in this process. It must be restored after subsequent startup of the device.
		Resume	Method An error state can be reset via this method. The "CurrentState" is reset to "Idle".

You can find a more exact description of these functions in the specifications "OPC Unified Architecture for AutoID" (Release 1.01), "OPC Unified Architecture Part 100: Devices" (Release 1.03) and the section "Application examples (Page 27)".

Application examples

3

This section describes typical applications. The main focus is on process and procedure descriptions. You can find an exact description of the associated variables, events, methods, data types and structures in the previous sections and the specifications "OPC Unified Architecture for AutoID" (Release 1.01) and "OPC Unified Architecture Part 100: Devices" (Release 1.03).

If these are elements whose contents or operating principles are not defined in the specification and that are extendable or relevant for the procedure description, they are described in the following sections.

3.1 Scan for transponders

The Ident devices have various functions for detecting/recording transponders in the antenna field.

3.1.1 "Scan" method

Method for synchronous execution of inventories or scanning of transponders. The method is called by the client, performs the scan and returns the detected transponders to the waiting client as result in the "Results" output parameter. "Results" is a field of structures of the type "RfidScanResults", whereby one structure is created per scanned transponder.

Structure

This structure contains, for example, the two elements "ScanData" and "CodeType" which need to be transferred as input parameters on addressed access to transponders, e.g. by the "ReadTag" method.

The "ScanData" element contains the EPC ID/UID of the scanned transponder. This element is defined as "Union" of the type "ScanData". The type used is defined or set during runtime by the "RuntimeParameters > CodeTypes" variable.

Table 3- 1 Supported data types

Value (ScanData)	Data type	Setting (CodeTypes)
1	ByteString	0
2	String	1
3	"ScanDataEpc" structure	2

The "CodeType" element of the "RfidScanResult" structure shows the data type of "ScanData" as string.

Possible values:

- RAW:BYTES
- RAW:STRING
- EPC

The other elements of the "RfidScanResult" structure contain additional values on the transponder scan.

A structure of the type "ScanSettings" must be transmitted to the "Scan" method as input parameter. This contains the abort criteria for the scan. The first criterion that is met ends the scan process.

Table 3- 2 Abort criteria

Criteria	Description
Cycles	Duration of the scan process based on performed scan cycles. 0 ≙ infinity
DataAvailable	Duration of the scan process • True: The scan process is ended when the first transponder is detected. • False: The scan process is ended by the stop criteria "Cycles" and "Duration".
Duration	Duration of the scan process in [ms] 0 ≙ infinity

Because the "Scan" method is a synchronous call, at least one of these abort criteria needs to be set.

3.1.2 "ScanStart" and "ScanStop" method

Method for asynchronous execution of inventories or scanning of transponders. The method is called by the client, starts the scan on the Ident device and returns.

The detected transponders are returned to the client asynchronously with events of the type "RfidScanEventType". The prerequisite for this is that the client logged on to the "RfidScanEvents" on the associated read point. An "RfidScanEvent" returns one or more transponders as a field of structures of the type "RfidScanResults", whereby one structure is created per scanned transponder.

Structure

This structure contains, for example, the two elements "ScanData" and "CodeType" which need to be transferred as input parameters on addressed access to transponders, e.g. by the "ReadTag" method.

The "ScanData" element contains the EPC ID/UID of the scanned transponder. This element is defined as Union of the type "ScanData". The type used is defined or set during runtime by the "RuntimeParameters > CodeTypes" variable.

Table 3- 3 Supported data types

Value (ScanData)	Data type	Setting (CodeTypes)
1	ByteString	0
2	String	1
3	"ScanDataEpc" structure	2

The "CodeType" element of the "RfidScanResult" structure shows the data type of "ScanData" as string.

Possible values:

- RAW:BYTES
- RAW:STRING
- EPC

The other elements of the "RfidScanResult" structure contain additional values on the transponder scan.

A structure of the type "ScanSettings" must be transmitted to the "StartScan" method as input parameter. This contains the abort criteria for the scan. The first criterion that is met ends the scan process.

Table 3- 4 Abort criteria

Criteria	Description
Cycles	Duration of the scan process based on performed scan cycles. 0 ≙ infinity
DataAvailable	Duration of the scan process • True: The scan process is ended when the first transponder is detected. • False: The scan process is ended by the stop criteria "Cycles" and "Duration".
Duration	Duration of the scan process in [ms] 0 ≙ infinity

Because the "StartScan" method works asynchronously, it is permitted to have no set abort criterion. The Ident device then scans "endlessly" or until the "ScanStop" method is called.

It is possible to end the scan process by calling the "ScanStop" method. The scan process is ended immediately in this case, event if a set abort criterion from "ScanSettings" is not yet fulfilled.

As soon as the scan process is ended, no more "RfidScanEvents" are sent to the client by the read point. If the login of the client to the "RfidScanEvents" is maintained at this read point, another call of "ScanStart" is sent from "RfidScanEvents" to the client.

3.1.3 "ScanActive" variable

Variable for asynchronous execution of inventories or for starting/stopping the scan process. Setting the "ScanActive =True" variable starts the scan for transponders; setting the "ScanActive = False" variable ends the scan. The detected transponders are returned to the client as contents of the "LastScanData" variable. The associated additional values can be queried by the client via the following variables:

- LastScanAntenna
- LastScanTimestamp
- LastScanRSSI

The "LastScanData" variable is defined as "BaseDataType". The type used is defined or set during runtime by the "RuntimeParameters > CodeTypes" variable.

Structure

Table 3- 5 Supported data types

Value (LastScanData)	Data type	Setting (CodeTypes)
1	ByteString	0
2	String	1
3	"ScanDataEpc" structure	2

The unwritable "RuntimeParameters > ScanSettings > CodeType" variable shows the currently set data type of the transponders in the "LastScanData" variable as readable string.

Possible values:

- RAW:BYTES
- RAW:STRING
- EPC

This method is only suitable for single tag mode and only when it can be ensured that the content of the variable is read by the client before the server overwrites the variables with the next values (see also section "3.1.1 Variables").

Associated with the "ScanActive" variable, there are the following variables with the abort criteria for the scan. The first criterion that is met ends the scan process.

Table 3- 6 Abort criteria

Criteria	Description
Cycles	Duration of the scan process based on performed scan cycles. 0 ≙ infinity
DataAvailable	Duration of the scan process • True: The scan process is ended when the first transponder is detected. • False: The scan process is ended by the stop criteria "Cycles" and "Duration".
Duration	Duration of the scan process in [ms] 0 ≙ infinity

Because the scan process via the "ScanActive" method works asynchronously, it is permitted to have no set abort criterion. The Ident device then scans "endlessly". By setting the "ScanActive = False" variable, you can end the scan process. The scan process is ended immediately in this case, event if a set abort criterion from "ScanSettings" is not yet fulfilled.

Note

Output of identified transponders

If the "RSSI events" parameter was enabled in the WBM in the "Communication > OPC UA" menu, the filtered out transponders are also reported.

3.1.4 Scans based on triggers

Independent asynchronous execution of inventories or scanning for transponders by the Ident device. This is only possible with the SIMATIC RF600 UHF readers. In this procedure, the reader must be configured accordingly via the WBM. Based on the configuration, the relevant read points scan for transponders as soon as this has been initiated by a defined trigger.

The detected transponders are returned to the client asynchronously with events of the type "RfidScanEventType". The prerequisite for this is that the client logged on to the "RfidScanEvents" on the associated read point. An "RfidScanEvent" returns one or more transponders as a field of structures of the type "RfidScanResults", whereby one structure is created per scanned transponder.

Structure

This structure contains, for example, the two elements "ScanData" and "CodeType" which need to be transferred as input parameters on addressed access to transponders, e.g. by the "ReadTag" method.

The "ScanData" element contains the EPC ID/UID of the scanned transponder. This element is defined as "Union" of the type "ScanData". The type used is defined or set during runtime by the "RuntimeParameters > CodeTypes" variable.

Table 3- 7 Supported data types

Value	Data type
0	ByteString
1	String
2	"ScanDataEpc" structure

The "CodeType" element of the "RfidScanResult" structure shows the data type of "ScanData" as string.

Possible values:

- RAW:BYTES
- RAW:STRING
- EPC

The other elements of the "RfidScanResult" structure contain additional values on the transponder scan.

As soon as the trigger condition is no longer met and the scan process is ended again, no more "RfidScanEvents" are sent to the client by the read point. If the login of the client to the "RfidScanEvents" is maintained at this read point and the trigger condition is met again, "RfidScanEvents" is sent to the client once more.

3.2 Access to transponders

In addition to scanning and detecting transponders, there are other options for accessing transponders. This includes all executable transponder commands, as defined in the specification "OPC Unified Architecture for AutoID" (Release 1.01) as synchronous OPC UA method calls. All transponder commands supported by the OPC UA servers of the SIMATIC Ident devices are listed below.

Note

Write operations to "EPC" memory bank with ID length "0"

When writing to the "EPC" memory bank, make sure that the length field of the EPC ID/UID is not accidentally set to "0". This has the result that the affected transponders can then no longer be addressed. In many cases, this amounts to a permanent total failure of the transponder. Therefore, if possible, the "WriteTagId" command should always be used instead of "WriteTag" to write to the "EPC" memory bank.

Supported functions

Table 3- 8 Supported functions

Functions	Description
KillTag	Destroy transponders. This function is only supported by UHF readers.
LockTag	Lock areas on the transponder. This function is only supported by UHF readers.
SetTagPassword	Set transponder-specific passwords. This function is only supported by UHF readers.
ReadTag	Read out transponder data.
WriteTag	Write transponder data.
WriteTagID	Write an EPC ID/UID including the Protocol Control Word. This function is only supported by UHF readers.

Example "ReadTag" (RF600)

The "ReadTag" method is addressed below as an example. This has the following signature:

```
ReadTag (
[in] ScanData Identifier
[in] CodeTypeDataType CodeType
[in] UInt16 Region
[in] UInt32 Offset
[in] UInt32 Length
[in] ByteString Password
[out] ByteString ResultData
[out] AutoIdOperationStatusEnumeration Status);
```

Input parameters

The input parameters "Region", "Offset" and "Length" specify which data should be read. These parameters must always be filled.

If, for example, reading should take place from memory bank 3 - this bank contains the user memory in UHF transponders in line with the ECPGlobal standard - as of address "0" to a length of 4 bytes, you need to set the input parameters "Region", "Offset" and "Length" as follows:

- Region = 3
- Offset = 0
- Length = 4

Via the two first input parameters "Identifier" and "CodeType", you can define whether the command should be called addressed or unaddressed. With an addressed command, this is applied to a specific transponder in a targeted way.

If the two parameters are not transmitted, the access is unaddressed. The command is then run on any transponder located in the antenna field. The prerequisite for this is that there is only exactly one transponder in the antenna field. If there is only one transponder in the antenna field, the user data read by the transponder is returned under the "ResultData" output parameter ("ByteString type).

Example: 11223344

Output parameters

If the command has been performed successfully, the "Status" output parameter returns the value "0" for "SUCCESS". The "AutoIdOperationStatusEnumeration" data type of the "Status" parameter is an enumeration that returns the status of an RFID command. This means that this output parameter returns the success status of the RFID processing of the command.

This parameter must not be confused with the OPC UA status code for the actual OPC UA method service which represents more basic errors (e.g. "Bad_MethodInvalid" for a method that is not supported).

If there is more than one transponder in the antenna field during unaddressed access, the "Status" output parameter returns the error status "9" for "MULTIPLE_IDENTIFIERS". In this case, the output parameter "ResultData" does not return any data (Length = -1, Content = 0).

If there is no transponder in the antenna field, the "Status" output parameter returns the error status "8" for "NO_IDENTIFIER". In this case, the output parameter "ResultData" does not return any data (Length = -1, Content = 0).

If only the selected transponder is in the antenna field during addressed access, the user data read by the transponder is returned under the "ResultData" output parameter ("ByteString" type) and the output parameter "Status" returns the value "0" for "SUCCESS".

If the transponder area to be read out is password-protected, the correct password must be transmitted via the "Password" input parameter. Otherwise, reading out the desired area is not possible. In this case, the output parameter "Status" returns the error status "3" for "PERMISSION_ERROR" and the output parameter "ResultData" does not return any data (Length = -1, Content = 0).

For addressed access to a specific transponder, the two input parameters "Identifier" and "CodeType" must be filled.

Structure

The "Identifier" input parameter contains the EPC ID/UID of the transponder to be accessed with the last command. This element is defined as "Union" of the type "ScanData". The following entries from this "Union" are supported by the Ident devices:

Table 3- 9 Supported data types

Value (ScanData)	Data type
1	ByteString
2	String
3	"ScanDataEpc" structure

The value of the "Identifier" input parameter then consists of the numbers 1, 2 or 3 to define the data type of the EPC-ID/UID.

The "CodeType" parameter has the "CodeTypeDataType" data type that was derived from the "String" data type. In this element, the data type defined for "Identifier" must be specified again as string.

Possible values:

- RAW:BYTES
- RAW:STRING
- EPC

Note

Transfer of the parameters

In terms of data type, the two input parameters "Identifier" and "CodeType" are identical to the elements "ScanData" and "CodeType" of an "RfidScanResult" structure (see also Result in section "Scan for transponders (Page 27)"). This means that it is possible to transmit the "ScanData" and "CodeType" elements of an "RfidScanResult" of the "ReadTag" method as input parameter "Identifier" or "ScanData" received through a previous scan.

3.3 Diagnostics

The OPC UA servers of the SIMATIC Ident devices have functions for diagnosing and tracking the plant via OPC UA.

3.3.1 Requirements

An OPC UA client can be connected to an Ident device as a main application or as a second application connected in parallel to an existing PROFINET, PROFIBUS, EtherNet/IP or XML main application for diagnostics purposes. Different conditions must be met depending on this.

OPC UA client as main application

For the diagnostics values to be available in the case of an OPC UA application as main application, the desired diagnostics options must be set in the WBM in the "Settings > Communication > OPC UA" menu. The following options are available:

- Presence events
- Last access events
- Log events

OPC UA client as secondary application

For an OPC UA client to also be able to connect in case of a PROFINET, PROFIBUS or EtherNet/IP main application, the "Parallel" parameter must be enabled in the WBM in the "Settings > Communication > OPC UA" menu.

This case is intended purely for diagnostics purposes, so only read access and no longer write access is possible here for OPC UA clients. In addition, the desired diagnostics options must be set in the WBM in the "Settings > Communication > OPC UA" menu. The following options are available:

- Presence events
- Last access events
- Log events

3.3.2 Retrieving diagnostic information

Diagnostics values are made available as variable values and as events. The prerequisite for this is that the desired diagnostics option (e.g. Presence-Events) is enabled.

Simple OPC UA clients restricted to the "DataAccess" OPC UA service can obtain the desired diagnostics values by reading or logging onto the corresponding diagnostics variable(s).

OPC UA clients with more functionalities should have the diagnostics values sent to them via the corresponding diagnostics events, if possible. For this purpose, the client needs to log on to the desired read point for these event types. After logon, every relevant diagnostics event

generated by this read point is sent to the client by the server. This takes place until the client logs off from these diagnostics events again.

3.3.3 "Presence" variable/event

Variable/event to display transponder presence. To receive presence information, the "Presence events" parameter must be set in the WBM in the "Settings > Communication > OPC UA" menu. Another prerequisite is that the respective Ident device supports "Presence" mode and is also operated in this mode.

Presence variable

Table 3- 10 Variable

Variable	Description
Diagnostics > Presence	Shows the presence of transponders. • 0: There is no transponder in the antenna field. • 1: There is at least one transponder in the antenna field. • >1: Exact number of transponders located in the antenna field

Presence event

An "AutoIdPresenceEvent" is generated for each change in presence. The "AutoIdPresenceEvent" contains the following OPC UA properties in addition to the properties inherited from "BaseEvent":

Table 3- 11 OPC UA properties

Property	Description
DeviceName	"DeviceName" (≙ DisplayName) of the read point that sent the event. This property is inherited from "AutoIdDiagnosticsEvent". The "BrowseName" of the read point is supplied via the inherited standard property "SourceName" of "BaseEvent".
Presence	Shows the presence of transponders. • 0: There is no transponder in the antenna field. • 1: There is at least one transponder in the antenna field. • >1: Exact number of transponders located in the antenna field

3.3.4 "Logbook" variables/event

Variable/event to read out log information. Header lines of the log can be queried using the variable. The last entry of the log can be queried or received via a variable or an event. This diagnostics functionality therefore offers many ways to read out information:

- Replicate the entire log of the device by appending the latest values in the OPC UA client.
- Write a specific error log by appending all read or received error entries in the OPC UA client.
- Write a specific transponder log by appending all read or received error entries about transponder access in the OPC UA client.

To receive log information, the "Log events" parameter must be set in the WBM in the "Settings > Communication > OPC UA" menu.

Note

Log covers all read points

Note that the log covers the information about the entire Ident device and thus the information for all read points. Consequently, the log contains all entries on all read points, irrespective of the read point via which the log information was queried using OPC UA.

You can define the read point via which the log information is queried in the WBM in the "Settings > Communication > OPC UA" menu.

Logbook variables

Table 3- 12 Variables

Variables	Description
Diagnostics - Logbook > LastLogEntry	The last entry in the log as formatted string Example: 2018-05-30T13:14:43.546+00:00 | COMMANDS | readTagIDs | CMD | 0 | Read-point_11 | 000000005575B67D | PC=0000 | RSSI=0 | Pwr=0
Diagnostics - Logbook > LogColumns	The headers column of the log as formatted string Example: Date_Time ; Type ; Entry ; Command_Type; Sequence_Number ; Read-point ; EPCID_UID ; PC ; RSSI ; Power

Logbook event

A "LastLogEntryEvent" is generated for every entry that was made in the log. The text of the log entry is in the "Message" OPC UA property inherited from the "BaseEvent".

Table 3- 13 Properties

Property	Description
Message	The last entry in the log as formatted string Example: 2018-05-30T13:14:43.546+00:00 | COMMANDS | readTagIDs | CMD | 0 | Read-point_11 | 000000005575B67D | PC=0000 | RSSI=0 | Pwr=0

The "LastLogEntryEvent" contains the following OPC UA property in addition to the OPC UA properties inherited from "BaseEvent":

Table 3- 14 OPC UA properties

Property	Description
DeviceName	"DeviceName" (≙ DisplayName) of the communications module that made the log entry and generated the event. This property is inherited from "AutoIdDiagnosticsEvent". The inherited standard property "SourceName" of "BaseEvent" contains the "BrowseName" of the read point via which the event was transmitted.

3.3.5 "LastAccess" variables/event

Variable/event to read out information on the last successful transponder access. Based on this information, a tracking system can be implemented, for example.

To receive log information, the "Last access events" parameter must be set in the WBM in the "Settings > Communication > OPC UA" menu.

Note

Output only of successful transponder access

Note that the "LastAccess" diagnostics only supplies information on successful transponder access. Information about failed transponder access is entered in the log as error entries and can be queried from there.

LastAccess variables

Note

Optional use of the variables as trigger for an OPC UA client

Please note that these variables supply logically related information. With successful transponder access, the contents of the "Timestamp" variable are written last. All other supported "LastAccess" variables are filled first. These variables can thus serve as a trigger for an OPC UA client in order to query the other variable values.

Depending on the real process, it may not be possible in some cases to ensure that the variables have been queried completely by the client in time before the next transponder access takes place and variables are written with new values again from the process. In this case, usage is not possible. Instead, use the "RfidAccessEvent" events, which are also supported by the device.

Table 3- 15 Variables

Variables	Description
Diagnostics - LastAccess > Client	Client interface via which the last transponder access on this read point took place. Possible values: • PNIO: Access via PROFINET • OPCUA: Access via OPC UA • WBM: Access via WBM • XML(1...4): Access via the XML channel • EIP: Access via EtherNet/IP • PB: Access via PROFIBUS • READPOINT: Internal access via the read point The read points of the device can scan for transponders independently. This is the case, for example, when at least one user application has started an asynchronous scan command or when a configured trigger was started with RF600. These events can also overlap, in which case they can no longer be clearly assigned. In this case, the value "READPOINT" is always used instead.
Diagnostics - LastAccess > Command	Command that was executed during the last transponder access operation. The output values depend on the client interface used:

Variables	Description	
	PNIO, PB, EIB (Ident profile): • INVENTORY • WRITE-ID • KILL-TAG • LOCK-TAG-BANK • PHYSICAL-READ • PHYSICAL-WRITE • FORMAT • TAG-STATUS • PUT	PNIO, PB (FB 45): • WRITE • READ • INIT • MDS-STATUS
	OPC UA: • Scan • ReadTag • WriteTag • KillTag • LockTag • SetTagPassword • Read • Write • WriteTagID	WBM, XML: • readTagIDs • writeTagID • readTagMemory • writeTagMemory • killTag • lockTagBank • getTagStatus
	READPOINT: • Observed	--
Diagnostics - LastAccess > Identifier	EPC ID/UID of the transponder which was accessed with the last command. Defined as "BaseDataType". The actual data type is defined during runtime by the "RuntimeParameters > CodeTypes" variable. The types "String", "ByteString" and "ScanDataEpc" are supported. Example (RF300 reader, data type string): 000000005575B67D	
Diagnostics - LastAccess > Timestamp	Time of the last transponder access Example: 2018-05-29T08:29:15.812Z	
Diagnostics - LastAccess > RWData	Read/written data of the last command. Only possible with commands for reading/writing the transponder memory. Defined as "BaseDataType". The actual data type is defined during runtime by the "CodeTypesRWData" variable. The types "String", "ByteString" and "Scan-DataEpc" are supported. Example (RF300 reader, data type string): 0011223344556677	

Variables	Description
Diagnostics - LastAccess > Antenna	Antenna via which the transponder was accessed with the last command. Possible values: Two-digit number. The first digit stands for the read point index, the second digit for the antenna index within the read point. Example: • Read point 1, antenna 1: 11 • Read point 2, antenna 1: 21
Diagnostics - LastAccess > CurrentPowerLevel	Power with which the last command was executed on the transponder. This function is only supported by UHF readers. Possible values: Decimal number for the power in [dB] Increment: 0.25 dB • 0 • 5.00 ... 33.00
Diagnostics - LastAccess > PC	Protocol Control Word of the transponder that was accessed last. This function is only supported by UHF readers. Example: 3000
Diagnostics - LastAccess > Polarization	Polarization with which the last command was executed on the transponder. This function is only supported by UHF readers. Possible values: • Default • Circular • Linear_vertical • Linear_horizontal
Diagnostics - LastAccess > Strength	RSSI value with which the last command was executed on the transponder. This function is only supported by UHF readers. Possible values: 0 ... 255

LastAccess event

The "RfidAccessEvent" is an event for reading out information on the last successful transponder access. It is generated for every command that was successfully executed on a transponder.

In addition to the inherited properties of "BaseEvent", an "RfidAccessEvent" also contains the following OPC UA properties:

Table 3- 16 OPC UA properties

Property	Description	
DeviceName	"DeviceName" (≙ DisplayName) of the read point that sent the event. This property is inherited from "AutoIdDiagnosticsEvent". The inherited standard property "SourceName" of "BaseEvent" contains the "BrowseName" of the read point via which the event was transmitted.	
Client	Client interface via which the last transponder access on this read point took place. Possible values: • PNIO: Access via PROFINET • OPCUA: Access via OPC UA • WBM: Access via WBM • XML(1...4): Access via the XML channel • EIP: Access via EtherNet/IP • PB: Access via PROFIBUS • READPOINT: Internal access via the read point The read points of the device can scan for transponders independently. This is the case, for example, when at least one user application has started an asynchronous scan command or when a configured trigger was started with RF600. These events can also overlap, in which case they can no longer be clearly assigned. In this case, the value "READPOINT" is always used instead.	
Command	Command that was executed during the last transponder access operation. The output values depend on the client interface used:	
	PNIO, PB, EIB (Ident profile): • INVENTORY • WRITE-ID • KILL-TAG • LOCK-TAG-BANK • PHYSICAL-READ • PHYSICAL-WRITE • FORMAT • TAG-STATUS • PUT	PNIO, PB (FB 45): • WRITE • READ • INIT • MDS-STATUS

Property	Description	
	OPC UA: • Scan • ReadTag • WriteTag • KillTag • LockTag • SetTagPassword • Read • Write • WriteTagID	WBM, XML: • readTagIDs • writeTagID • readTagMemory • writeTagMemory • killTag • lockTagBank • getTagStatus
	READPOINT: • Observed	--
AccessResult	The property "AccessResult" is a field of "RfidAccessResult" structures. A field entry or a structure carries the information for a transponder that was accessed by the "Command". If multiple transponders were accessed by the command, the field has multiple entries. The properties of an "RfidAccessResult" are explained below.	

Table 3- 17 Design of the "RfidAccessResult" structure

Property	Description
Identifier	EPC ID/UID of the transponder which was accessed with the last command. Defined as "BaseDataType". The actual data type is defined during runtime by the "RuntimeParameters > CodeTypes" variable. The types "String", "ByteString" and "ScanDataEpc" are supported. Example (RF300 reader, data type string): 000000005575B67D
CodeType	Shows the set data type of "Identifier" as a string. Possible values: • RAW:BYTES • RAW:STRING • EPC
Timestamp	Time of the last transponder access Example: 2018-05-29T08:29:15.812Z
RWData	Read/written data of the last command. Only possible with commands for reading/writing the transponder memory. Defined as "BaseDataType". The actual data type is defined during runtime by the "CodeTypesRWData" variable. The types "String", "ByteString" and "Scan-DataEpc" are supported. Example (RF300 reader, data type string): 0011223344556677

Property	Description
CodeTypeRWData	Shows the set data type of "RWData" as a string. Possible values: • RAW:BYTES • RAW:STRING • EPC
Antenna	Antenna via which the transponder was accessed with the last command. Possible values: Two-digit number. The first digit stands for the read point index, the second digit for the antenna index within the read point. Example: • Read point 1, antenna 1: 11 • Read point 2, antenna 1: 21
CurrentPowerLevel	Power with which the last command was executed on the transponder. This function is only supported by UHF readers. Possible values: Decimal number for the power in [dB] Increment: 0.25 dB • 0 • 5.00 ... 33.00
PC	Protocol Control Word of the transponder that was accessed last. This function is only supported by UHF readers. Example: 3000
Polarization	Polarization with which the last command was executed on the transponder. This function is only supported by UHF readers. Possible values: • Default • Circular • Linear_vertical • Linear_horizontal
Strength	RSSI value with which the last command was executed on the transponder. This function is only supported by UHF readers. Possible values: 0 ... 255

3.4 Firmware update

The firmware update is realized according to the Standard OPC UA SoftwareUpdate model of the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03). This standard describes a model for updating software or firmware.

3.4.1 Query of current version

The version of the currently installed firmware can be queried by querying the value of the OPC UA property "SoftwareUpdate > Loading > CurrentVersion > SoftwareRevision".

Example: V04.00.00.00_xyz

3.4.2 Query of pending version

The version of the pending firmware can be queried by querying the value of the OPC UA property "SoftwareUpdate > Loading > PendingVersion > SoftwareRevision".

Example: V04.00.00.00_xyz

If the property is empty, no firmware has been transferred to the device for activation yet.

3.4.3 Boundary conditions when transferring firmware

The following OPC UA variables or properties specify boundary conditions when transferring the firmware or provide assistance during the transfer.

- SoftwareUpdate > Loading > WriteBlockSize

  This property shows the maximum permitted size of a memory block when transferring the firmware. The maximum value is always 1 048 576 bytes, which corresponds to 1 024 KB. The firmware of the Ident devices must be transferred in blocks with this maximum size. Larger blocks lead to an error.

- SoftwareUpdate > Loading > FileTransfer > ClientProcessingTimeout

  Maximum wait time that the server accepts when transferring between the individual firmware blocks. If this time is exceeded by the client, the server considers the transfer to have failed. The time span is always 20 000 ms or 20 s.

- SoftwareUpdate > Loading > ErrorMessage

  If an error has occurred during loading, this variable shows a more detailed description of the error.

3.4.4 Transferring firmware to the device

If no firmware has been transferred to the device for activation yet or if it does not correspond to the desired version, firmware must be transferred to the device before it can be activated. This is done using the file operation "SoftwareUpdate > Loading > FileTransfer".

This operation has the "TemporaryFileTransfer" type and provides the mechanisms to create a temporary "FileType" object, which in turn provides the file operations for actual writing or transfer. This temporary "FileType" object is created by calling the method "SoftwareUpdate > Loading > FileTransfer > GenerateFileForWrite".

Input/output parameter of the method

The value 1 ("Pending") must always be transmitted as values for the "GenerateOptions" input parameter. Other values are not accepted.

The following values are returned by the method as output parameters:

- FileNodeId (always 100 000)

  This is the "NodeId" of the created temporary "FileType" object. Like all "NodeIds", this "NodeId" is also necessary for addressing. It must be used to address the temporary "FileType" object to be able to call the methods for the file operations on this object.

- FileHandle

  This value must be specified as input parameter for all file operations via the temporary "FileType" object.

After successful execution of the method, the temporary "FileType" object exists. If a "Rebrowse" of the address area is performed in a generic client, there is now a "TemporaryFile" node under "SoftwareUpdate – Loading – FileTransfer". This represents this temporary "FileType" object.

This object makes methods available for file operations. Only the "Write" method is relevant for the firmware update. The firmware can be transferred to the device block-by-block by repeated calls of the "Write" method. The "FileHandle" received by the "GenerateFileForWrite" method must be supplied to the "Write" method as input parameter.

Note

Conditions

Note that the following conditions must be observed when transferring the firmware:

- The block size must not exceed the value 1 048 576 bytes (1024 KB).
- The interval between the individual transferred firmware blocks must not be greater than 20000 ms (20 s).

Completing the transfer

After the firmware has been completely transferred to the device, you need to complete the transfer. This is done by calling the method "SoftwareUpdate > Loading > FileTransfer > CloseAndCommit".

The "FileHandle" must be supplied to this method as input parameter. The temporary "FileType" object is destroyed again. If a "Rebrowse" of the address area is performed subsequently in a generic client, the "TemporaryFile" node attached temporarily under "SoftwareUpdate > Loading > FileTransfer" is removed again.

If no error has occurred, the firmware is now completely available as "PendingVersion" on the Ident device. This can be checked by querying the pending version "SoftwareUpdate > Loading > PendingVersion > SoftwareRevision". This OPC UA property should now have assumed the version of the transferred firmware.

If an error has occurred during transfer of the firmware, you can refer to the variable "SoftwareUpdate > Loading > ErrorMessage" for a more detailed error description.

3.4.5 Activation of transferred firmware

To complete a firmware update, the transferred firmware still needs to be activated. This can take place at any later time.

Before the firmware activation, query the current state of the activation. This is done by querying the value of the OPC UA property "SoftwareUpdate > Installation > CurrentState".

This property can have the following values:

- 1: Idle
- 2: Installing
- 3: Error during installing

If the "Idle" value is output, you can continue with the activation directly. In contrast, the "Installing" value shows that firmware activation is currently being performed (writing firmware to the flash). In this case, you cannot continue with the activation until the current activation has been completed. If the "Error" value is output, this indicates that an error occurred during the last firmware activation. You can acknowledge or reset the error state by calling the method "SoftwareUpdate > Installation > Resume". The value then changes to "Idle" and you can continue with the activation.

The firmware is activated by calling "SoftwareUpdate > Installation > InstallSoftwarePackage".

Parameterization of the "InstallSoftwarePackage" method

This method requires multiple input parameters. The "ManufacturerUri" and "SoftwareRevision" parameters must be specified. Enter the corresponding values of the firmware that has been transferred but not yet activated ("Pending"). No other parameters need to be filled.

- ManufacturerUri

  In this parameter, you need to specify the value of the variable "SoftwareUpdate > Loading > PendingVersion > ManufacturerUri" (https://www.siemens.com).

- SoftwareRevision

  In this parameter, you need to specify the value of the variable "SoftwareUpdate > Loading > PendingVersion > SoftwareRevision".

The call starts the activation of the firmware, i.e. the writing of the firmware to the flash memory.

Note

Duration of the firmware update

Note that writing the firmware to the flash memory takes several minutes. "InstallSoftwarePackage" is a synchronous call. This means that the method is ended with an error ("BadTimeout") because the device is performing a restart. If a problem occurs as a consequence of the call, the corresponding error is reported.

Note

Restore the OPC UA connection after the restart

Immediately after successful firmware activation, the Ident device performs a restart independently. Note that the OPC UA connection is interrupted and needs to be restored after startup of the device.

Checking the activation

After startup of the device and restore of an OPC UA connection, you should check the firmware activation again.

After successful activation, the value of the OPC UA property "SoftwareUpdate > Loading > PendingVersion > SoftwareRevision" should be empty. The value of the property "SoftwareUpdate > Loading > CurrentVersion > SoftwareRevision" should now correspond to the version of the most recently activated firmware. If this is not the case, an error has occurred during activation of the firmware and the process needs to be repeated.

3.5 Backup/restore of the configuration

The OPC UA servers of the SIMATIC Ident devices provide the possibility to detect configuration changes of the device and to read or write the configuration of the device. This is done using the following variables:

- RuntimeParameters > Configuration > ConfigID

  This variable can only be written and contains a unique identifier of the configuration saved in the device as alphanumeric text from hexadecimal characters.

  Example: 5FAAC0DE

  If the value of this variable changes, the saved configuration of the device has been changed (e.g. via the WBM of the device) or the device has been replaced. The value of this variable is automatically monitored by the OPC UA client to be able to detect a possible change to the configuration at any time.

- RuntimeParameters > Configuration > ConfigData

  This readable and writable variable contains the current, complete configuration of the Ident device as alphanumeric text in XML format. By reading the value of this variable, the current configuration of the device can be retrieved and saved in the client. By setting the value of this variable, the saved configuration is transferred to the device and activated.

Device replacement

During a device replacement, you can automatically transfer the configuration of the old device to the new device with these variables. To detect that the configuration of the device has been changed, the OPC UA client monitors the value of the variable "RuntimeParameters > Configuration > ConfigID". If the value of this variable changes, this indicates to the client that the device has been replaced. In this case, the client sets the previously read and saved configuration of the old device as a new value of the variable "RuntimeParameters > Configuration > ConfigData" of the new device.

The written configuration is activated at the same time as the write process to this variable. The new device now contains the configuration of the old device.

The following object is located on the same level as these two variables:

- RuntimeParameters > Configuration > Configuration

  This has the type "SoftwareType" and, in later versions of the Ident devices, will serve to realize the backup/restore of the device configuration according to the standard OPC UA SoftwareUpdate model from the specification "OPC Unified Architecture Part 100: Devices" (Release 1.03). This function cannot be used yet with the current firmware version.

Service & Support

A

Industry Online Support

In addition to the product documentation, you are supported by the comprehensive online information platform of Siemens Industry Online Support at the following Internet address:
Link: (https://support.industry.siemens.com/cs/de/en/)

Apart from news, you will also find the following there:

- Project information: Manuals, FAQs, downloads, application examples etc.
- Contacts, Technical Forum
- The option to submit a support request:
  Link: (https://support.industry.siemens.com/My/ww/en/requests)
- Our service offer:

  Right across our products and systems, we provide numerous services that support you in every phase of the life of your machine or system - from planning and implementation to commissioning, through to maintenance and modernization.

You will find contact data on the Internet at the following address:
Link: (https://www.automation.siemens.com/aspa_app/?ci=yes&lang=en)

"Industrial Identification" homepage

You can find the latest general information about our identification systems on the Internet at our Homepage (www.siemens.com/ident).

Online catalog and ordering system

The online catalog and the online ordering system can also be found on the Industry Mall home page (https://mall.industry.siemens.com).

SITRAIN - Training for Industry

The training offer includes more than 300 courses on basic topics, extended knowledge and special knowledge as well as advanced training for individual sectors - available at more than 130 locations. Courses can also be organized individually and held locally at your location.

You will find detailed information on the training curriculum and how to contact our customer consultants at the following Internet address:

Link: (https://new.siemens.com/global/en/products/services/industry/sitrain.html)