OPC Unified Architecture Data Access Client

OPC Unified Architecture Data Access Client communication protocol

Protocol specification

The OPC UA protocol is the next generation of the OPC standard, which provides a cohesive, secure, and reliable platform-independent framework for access to the real-time data. The main difference in comparison with the previous versions is that the interprocess communication uses TCP/IP communication instead of COM/DCOM technology and therefore is OS (Windows) independent. This protocol supports two methods of data encoding (binary and XML). The existing OPC client implementation supports only binary encoding.

Basic concepts

Endpoint: a physical address on a network, which allows the client to access one or more of the services provided by the server.

Session: it is an abstract connection of the OPC UA server and a client on the OSI application layer.

Monitored item: an entity on the OPC UA server that is defined by the client. It is used for monitoring the values.

Subscription: an object on the OPC UA server that is defined by the client. It returns the notifications about the change of monitored items.

Initiation of communication

When initiating the communication, the client exchanges multiple messages with the server. In the case of binary encoding, **"Hello Message"** is the first message sent from the client to the server. The message defines the size of receiving and sending buffers and the maximum size of messages that may be exchanged during TCP communication between client and server. It also defines the URL address of the endpoint. The server answers by sending an **"Acknowledge message"**, in which it confirms the suggested parameters or modifies them according to its limits.

The next message sent by the client is **"OpenSecureChannel message"**. It is used to establish a communication channel to exchange data. In this message, the client and server agree on what type of encrypting and signing mode will be used (either "sign and encrypt" or "encrypt" only). OPCUA client in D2000 System supports only unencrypted mode.

After establishing the communication channel, the client can send the message to create a session, **"CreateSession Message"**. It is the connection on the OSI application layer. After the server confirms the request, the session must be activated by **"ActivateSession Message"**. In this message, the client and server will agree on the algorithm for signing and encrypting if this mode has been agreed on when establishing the communication channel.

After activating the session, all types of messages that are intended for object management in the address space of the OPC UA server can be swapped. In this step, the client creates a subscription with the parameters set on the communication station level for all stations within the communication line.

The monitored items should be then inserted into these subscriptions. They correspond with the instances of I/O tags, which contain the parameters defined in the address dialog window.

From this moment, the server informs the client about the changes in the monitored objects by **"Publish message"** in the periodic intervals (that have been set in the parameters of subscription). If the monitored objects have not been changed, the server will send a **"Publish message"** once in (Max KeepAlive Count * Publishing Interval) seconds. The message informs the client that the subscription is still active. A similar check mechanism is also on the client-side - it will send a **"Publish message"** once every (Max KeepAlive Count * Publishing Interval). If the client does not send the acknowledgment **"Publish message"** within (LifeTime Count * Publishing Interval), the subscription will expire on the side of the server.

OPC UA communication was tested with:

- Simatic S-7 OPC UA Server
- Bernecker PLC embedded OPC UA Server
- Zenon OPC UA Server

Communication line configuration

Communication line category: OPC UA Client
Host address: OPC UA server address. You may set the name according to UNC convention (e.g. "\\server" or "server", DNS names (e.g. "domain.com", "example.company.com") or IP address ("196.54.23.113"). In the case of redundant systems, multiple names/addresses separated by commas or semicolons can be entered.
TCP port: TCP port of OPC UA server (e.g. 4840).
EndpointUrl: Endpoint address (e.g. *opc.tcp:///localhost:4840*)
Encoding type: Type of encoding that is used for data exchange (currently only *Binary encoding* is supported).

Protocol configuration on the communication line

Parameter name	Meaning	Unit	Default value
Client Type	Type of used client (driver for OPC UA communication): • Default - an original implementation of OPC UA client. Supports authentication (parameter Authentication Type) of Anonymous/Username types only. It does not support message encryption or signing. • Secure - a new implementation of OPC UA client with security support. Supports advanced authentication, encryption, and message signing capabilities.	Default Secure	Default
Session Name	Session text identifier. Session identifier should be unique within the client instance, making it possible to search problems faster in the client or server logs.	String	Kom process
Requested Channel Lifetime	The channel must be reopened before this time limit elapses. If the time is exceeded, the channel will be closed and unable to change data.	hh:mm:ss	01:00:00
Requested Session Timeout	Any message should be changed between client and server before this time limit elapses. If it is not sent, the sources within the session that are kept on the server are released. The primary work of this parameter is to remove the sessions that became inactive because of some unexpected reason.	mm:ss	01:00
Authentication Type	Type of authentication used with the OPC UA server. Supported types are: • Anonymous: logon is anonymous • Username: logon uses user name and password • Certificate: logon uses x509 certificate (only for Client Type = Secure)	Anonymous / Username	Anonymous
Token User Name	If Authentication type = Username, then user name used in the authentication. If Authentication type = Certificate, then path to user certificate (e.g. D:\user_cert.der).		
Token Password	If Authentication type = Username, then password used in the authentication. If Authentication type = Certificate, then path to user private key (e.g. D:\user_private_key.pem).		
Security Policy	Security policy (only for Client Type = *Secure*; for Client Type = *Default* a security policy *None* is used): • *None* - security policy None • *Basic128Rsa15* - security policy Basic128Rsa15 (considered to be obsolete due to using a weak SHA-1 hashing algorithm) • *Basic256* - security policy Basic256 (considered to be obsolete due to using a weak SHA-1 hashing algorithm) • *Basic256Sha256* - security policy Basic256Sha256 • *Aes128Sha256RsaOaep* - security policy Aes128Sha256RsaOaep • *Aes256Sha256RsaPss* - security policy Aes256Sha256RsaPss	None Basic128Rsa15 Basic256 Basic256Sha256 Aes128Sha256RsaOaep Aes256Sha256RsaPss	None

SecurityMode	A mode of message security in OPC UA communication (only for Client Type = *Secure*; for Client Type = *Default* a mode of message security *None* is used): • *None* - messages are not secured • *Sign* - messages are signed (protected against modification, but not against eavesdropping) • *Sign & Encrypt*- messages are signed and encrypted (protected both against modification and eavesdropping)	None Sign Sign & Encrypt	None
Reconnect Delay	Waiting after the connection is broken before the connection is re-established.	mm:ss.mss	00:10.000
Error Connect Delay	Waiting after an unsuccessful connection attempt.	mm:ss.mss	00:02.000
Debug Mode	It changes the number of information about communication. We recommend enabling the Extended/Full modes only when detecting the problems and debugging the communication. The "Full + Trace (Secure only)" mode is valid only for Client Type = Secure.	Normal /Extended/Full/ Full + Trace (Secure only)	Normal
Debug Threads	The parameter defines the thread(s) that will send the debug info about the communication.	Receiving /Sending/Others treads/All threads	All threads

Note: all X509 certificates used in OPC UA communication can be found in the following subdirectories of the *kom-opcua* directory in the application directory:

- *own* - a directory with KOM process's own certificate (file *cert.der*). If this file does not exist, it is generated
  Warning - this automatically generated certificate will only be valid for 1 year, so we recommend replacing it with a certificate valid for a longer period!
- *private* - a directory with a private key for the KOM process own certificate (file *private.pem*)
- *rejected* - a directory with rejected certificates
- *trusted* - a directory with trusted certificates (the first time a connection is established to an OPC UA server, its certificate is stored in this directory)

Protocol configuration on the communication station

The parameters on the level of the communication station correspond with the setting of one subscription. It means the one communication station is equivalent to one instance of a subscription within the session.

Full name	Description	Unit	Default value
Requested Publishing Interval	Defines the time interval for the server to send the information about the change of monitored items within the instance subscription by **"Publish message"**. Note: This parameter defines a proposed value that the OPC UA server can change, e.g. Bernecker-Rainer always returned a value of "Publishing Interval" at least 50 ms, although the requested interval was smaller.	mi:ss.mss	00:05.000
Requested LifeTime Count	If the client does not send the request for data till the time defined by (LifeTime Count * Publishing Interval), the subscription expires. The value should be minimally 3 times higher than the "Requested Max KeepAlive Count". Note: This parameter defines a proposed value that the OPC UA server can change, e.g. Bernecker-Rainer always returned as a value of "LifeTime Count" a maximum of 600, although the requested value was greater.	Number	1000
Requested Max KeepAlive Count	If the objects of subscription are not changed, the server will send a keep-alive message after elapsing the time (Max Notifications Per Publish * Publishing Interval). The client will confirm this message when it sends a new request for data. Note: This parameter defines a proposed value that the OPC UA server can change, e.g. Bernecker-Rainer has always returned as a value of "Max KeepAlive Count" a maximum of 200, although the requested value was greater.	Number	5
Max Notifications Per Publish	The parameter defines the maximum number of notifications about the object change, which the server can send in one **"Publish message"**. Zero indicates that the number of notifications is unlimited.	Number	0
Publishing Enabled	The parameter enables/disables the publishing within the subscription.	YES /NO	YES
Priority	It defines a relative priority of a subscription. If the server should send more notifications, the subscription with higher priority is preferred.	0-255	0
Samples Queue Size	This parameter enables creating an object queue with the defined length on the OPC UA server's side for each monitored item in a subscription.	Number	0
Read Timestamp	Timestamps used while reading a value: • None - timestamp received from OPC server is not used (value will be timestamped by the current time) • Source - *SourceTimestamp* is used • Server - *ServerTimestamp* is used (default)	-	Server

Write Timestamp	Timestamps used to write a value: • None - no timestamp is set • Source - *SourceTimestamp* is set (equal to the timestamp of written value) • Server - *ServerTimestamp* is set (equal to current time) • Both - both *SourceTimestamp* and *ServerTimestamp* are set Note: If the OPC server does not support the writing of timestamps, according to the standard it should return the *Bad_WriteNotSupported* (2155020288) error code.	-	None
Write Status Code	*StatusCode* item will be used when writing. Note: According to the standard, the OPC UA Wrapper returns the *Bad_WriteNotSupported* (2155020288) error code if the StatusCode entry is used when writing to the OPC DA Server version 2.05a.	YES /NO	YES
Write Whole Array	When an item of an array is written, the entire array is read first and then written. If this parameter is set to NO, only a specific array item is written, Note: According to the standard, if the OPC server does not support writing a specific array item, it should return the *Bad_WriteNotSupported* (2155020288) error code. Note: If this parameter is active, the "*Write only*" parameter must not be set at the I/O tag which addresses an item of the array.	YES /NO	NO
Read Mode	A way of reading values: • *Subscribe*: communication via subscriptions and notifications (standard) • *Subscribe + Read*: like *Subscribe*, in addition, periodic *Read* requests are sent (according to the time parameters of the station) • *Read*: send *Read* requests only Note: The *Subscribe+Read* and *Read* modes should only be used if there is a problem with standard communication, as they are less efficient and have a higher overhead.	Subscribe Subscribe+Read Read	Subscribe
No Filter	Ignoring filter parameters in the I/O tag configuration (Sampling type, DeadBand type, Trigger type). In the specific case, the OPC UA server did not work correctly if monitored items with the specified filter parameters were inserted into the subscription.	YES /NO	NO

I/O tag configuration

I/O tag configuration dialog window is used for setting the monitored objects.

Object address setting

Name	Meaning	Unit	Default value
ID	The identifier in text format, which is, in dependence on **ID type,** converted to the required native type. Note: if an identifier **%IGNORE** is specified for *ID type=String*, the I /O tag is ignored.	String	
ID type	Enumerated types of identifiers. They help to access the objects in OPC UA address space. *Numeric-1B ID*: Identifier limited to 1-byte value (0-255) *Numeric-2B ID*: Identifier limited to 2-byte value (0-65535) *Numeric-4B ID*: 4-byte identifier *String*: Text identifier *Guid -16B ID*: 16-byte (128-bit) number that is usually divided into four parts. For example 3F2504E0-4F89-11D3-9A0C-0305E82C3301. *ByteString*: Identifier that is represented as a sequence of bytes.	Numeric-1B ID / Numeric-2B ID/ Numeric-4B ID/String/Guid -16B ID /ByteString	Undefined
Namespace	Numerical identifier of the namespace of OPC UA server. Each OPC UA server can have N namespaces. However, the object identifier must be unique in one namespace.	Numeric	
Variable type	The value type of objects that can be processed by OPC UA client. *Variable type* should be used only if the I/O tag is intended for writing. As regards the reading of the object value, the information about type is sent together with the value.	Undefined / Boolean / Byte / SByte / Integer16 / Unsigned16 / Integer32 / Unsigned32 / Integer64 / Unsigned64 / Float / Double / String / UTC Time / Boolean array / Byte array / SByte array / Integer16 array / Unsigned16 array / Integer32 array / Unsigned32 array / Integer64 array / Unsigned64 array / Float array / Double array / String array / UTC Time array / LocalizedText / LocalizedText array	Undefined

Array index	If the object value is represented as a value array *(Boolean array / Byte array / SByte array / Integer16 array / Unsigned16 array / Integer32 array / Unsigned32 array / Integer64 array / Unsigned64 array / Float array / Double array / String array / UTC Time array)*, the parameter defines its range or value of a particular item. The first element of array is identified by index 0. A text representation of array index may be in several formats: • Separate integer, e.g. **"6"** or **"0":**- when you want to obtain a single value from the array. • Two integers separated by a colon, e.g. **"6:7"**, - if you want to obtain the range of values. • The expression separated by a comma in the case of a multidimensional array, e.g. **"6,7"** - when you want to obtain the particular value of the item of a 2D array. If you want to define a range, you should use expressions separated by a comma, e.g. **"6:8,7:10"**. Note: Writing is only supported for I/O tags with a specific index, not for ranges.	String	
Write only	It controls if the I/O tag is a part of the subscription. Its value will be sent periodically from the server in "Publish message".	Unchecked/checked	Unchecked
Expanded Node ID	If it is checked, it enables addressing the ExpandedNodeId. Unlike the classic identifier in the OPC UA address space, ExpandedNodeId is supplemented by **NameSpace URI** and **Server index.** Note: ExpandedNodeId is not yet supported in the KOM process.	Unchecked/checked	Unchecked
NamespaceUri	Text identifier of the namespace of the OPC UA server that is used instead of the numerical representation of a namespace.	String	
ServerIndex	A numerical identifier that addresses the server number when using the ExpandedNodeID identifier.	Numeric	0

Settings of other parameters

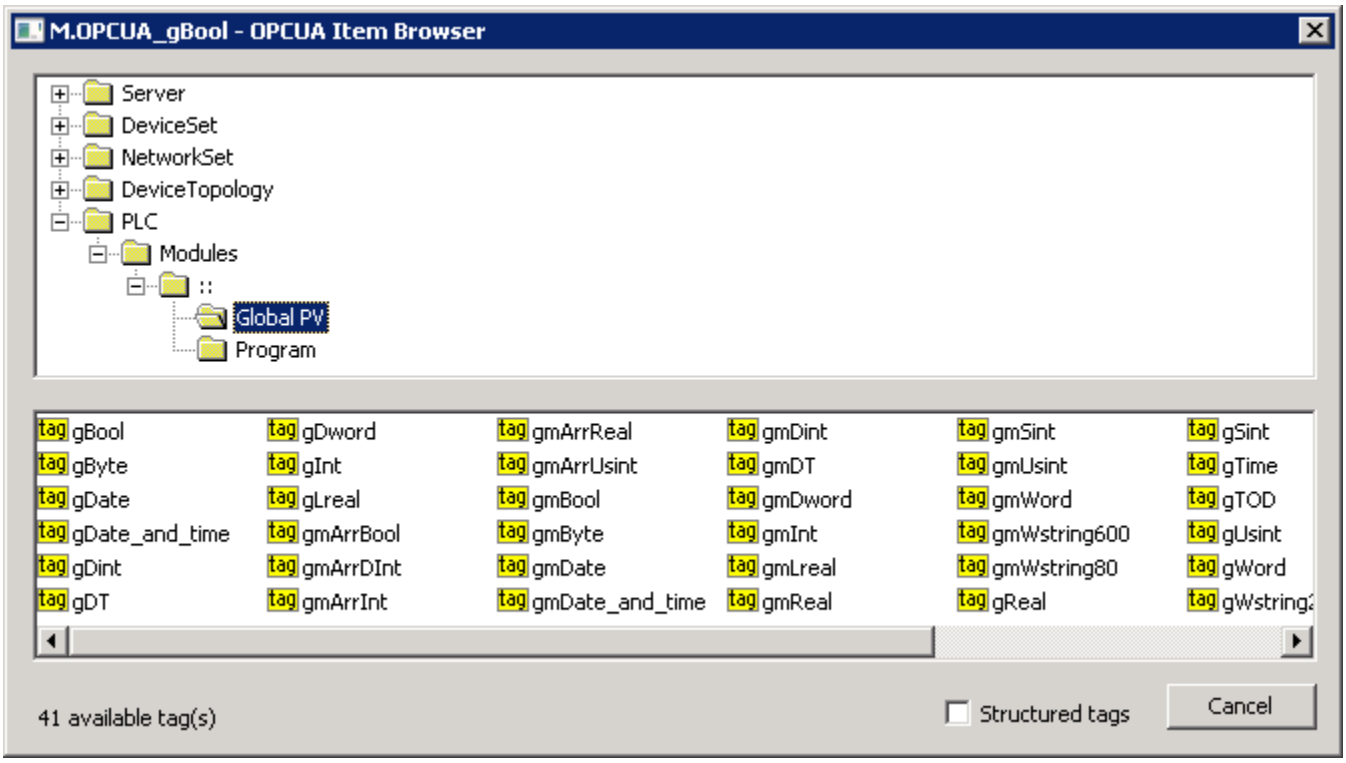
Name	Meaning	Unit	Default value
Sampling type	The parameter defines a sampling frequency of the monitored objects. When using the "Publishing rate", the frequency is equivalent to time Requested Publishing Interval, which is set on the communication station level. "Practical fastest rate" sets the sampling frequency on the maximum value. "Custom rate" enables to specify the custom sampling interval, which may be defined in "Sampling Time".	Publishing rate /Practical fastest rate /Custom rate	Publishing rate
Sampling time	The parameter allows you to set the custom sampling frequency if "Sampling type" is "Custom rate".	ss.ms	0.0
DeadBand type	Deadband is a band in which the change of value does not cause Data Change Notification, which is the part of Publish Message. When using "None", this band is ignored. Otherwise, there is used the relative or absolute value ("Percent"/" Absolute") from "DeadBand value".	None/Absolute /Percent	None
DeadBand value	The parameter defines the custom value of a deadband if you chose the relative/absolute value ("Percent"/"Absolute").		0.0
Trigger type	The parameter specifies the condition which causes Data Change Notification. When using "Status", only the status change is reported. Change of value and timestamp are ignored. When using "Status,Value", the change of timestamp is ignored. "Status,Value,Timestamp" ensures the reporting in all options, i.e. when changing the status, value, or timestamp. Note: a specific Simatic S7-1500 did not send value changes if this parameter was set to default "Status, Value, Timestamp" - changing it to "Status, Value" helped.	Status/Status, Value/Status, Value, Timestamp	Status, Value, Timestamp

Browser dialog window

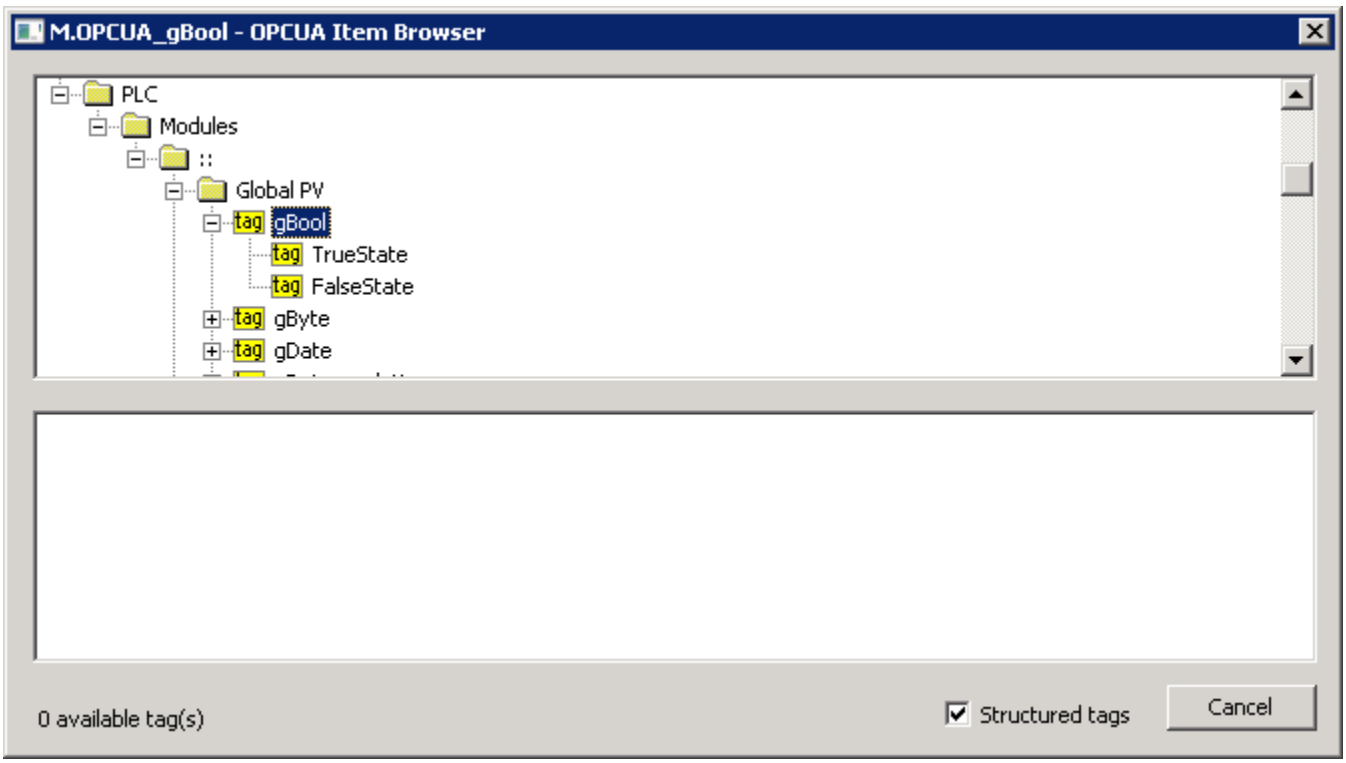
This dialog window is intended for browsing and inserting the OPC UA objects into the address parameter of the I/O tag. The upper part contains the tree structure of the address space. When clicking on the object, the lower part of the window displays the direct descendants of the object (variables, tags). Double click on one of the descendants transfers the address parameters of an object to the address dialog window of the I/O tag.

Note: Using Ctrl+C it is possible to copy a list of displayed descendants into the Windows clipboard. All descendants will be copied unless a specific descendant is selected.

Note: In versions from 17th December 2018 and newer, the recycling of browser dialog has been implemented. If the dialog is closed by the Close button or after selecting a tag, it is actually only hidden and it is available for browsing by another I/O tag within the same station so that the tree structure of the browsed objects is preserved. Clicking on the close icon at the top right corner will cause the dialog to be really closed.



Checking the "Structured tags" option causes the variables (tags) to appear in the tree structure in addition to the objects, and the KOM process also attempts to read their descendants. This is useful for browsing OPC UA servers that support structured tags. You can also insert a tag into the address dialog window of the I/O tag by double-clicking the tag name in the tree structure.



Tell commands

Command	Syntax	Meaning

STWATCH	STWATCH StationName	Tell command sends commands for the reading of values of all configured I/O tag

Literature

OPC Foundation manuals are placed on http://www.opcfoundation.org.

- OPC UA Part 1 - Overview and Concepts 1.01 Specification
- OPC UA Part 2 - Security Model 1.01 Specification
- OPC UA Part 3 - Address Space Model 1.01 Specification
- OPC UA Part 4 - Services 1.01 Specification
- OPC UA Part 5 - Information Model 1.01 Specification
- OPC UA Part 6 - Mappings 1.00 Specification
- OPC UA Part 7 - Profiles 1.00 Specification
- OPC UA Part 8 - Data Access 1.01 Specification

Changes and modifications

- May 10, 2012 - creating the document

Document revisions

- Ver. 1.0 – May 10, 2012
- Ver. 1.1 - December 17, 2018: Added browser dialog recycling and browsing of structured tags

Related pages:

Communication protocols