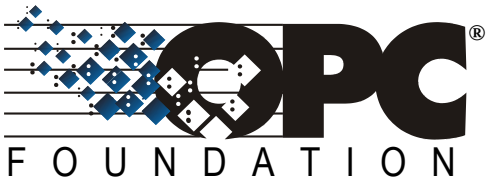


Joined Technical Specification

PLCopen and OPC Foundation

OPC-UA Client FUNCTION BLOCKS for IEC61131-3

Release 1.0 published

Date: April 03, 2014

Total number of pages: 34

The following paper

OPC-UA Client FUNCTION BLOCKS for IEC61131-3

is a joined document from PLCopen and OPC Foundation.

It summarises the results of the PLCopen OPC UA Task Force, containing contributions of all its members.

Armin Hornung	3S-Smart Software Solutions GmbH
Matthias Maier	3S-Smart Software Solutions GmbH
Matthias Damm	Ascolab GmbH
Stefan Hoppe	Beckhoff Automation GmbH, Chairman
Puja Lümkemann	Beckhoff Automation GmbH
Henning Mersch	Beckhoff Automation GmbH
Uwe Köhler	Bosch Rexroth AG
Stefan Benkner	Robert Bosch GmbH
Stefan Stemp	B&R Industrie-Elektronik GmbH
Karl Mayr	B&R Industrie-Elektronik GmbH
Bernd Schäfer	HIMA
Rene Simon	Hochschule Harz, PLCopen Board
Andreas Weichelt	KW-Software GmbH
Eelco van der Wal	PLCopen
Ingo Weber	Siemens AG

Change Status List:

Version number	Date	Change comment
V 0.9.8	3.04.2014	Release Candidate – as discussed in the web meeting
V 1.0	3.04.2014	Released

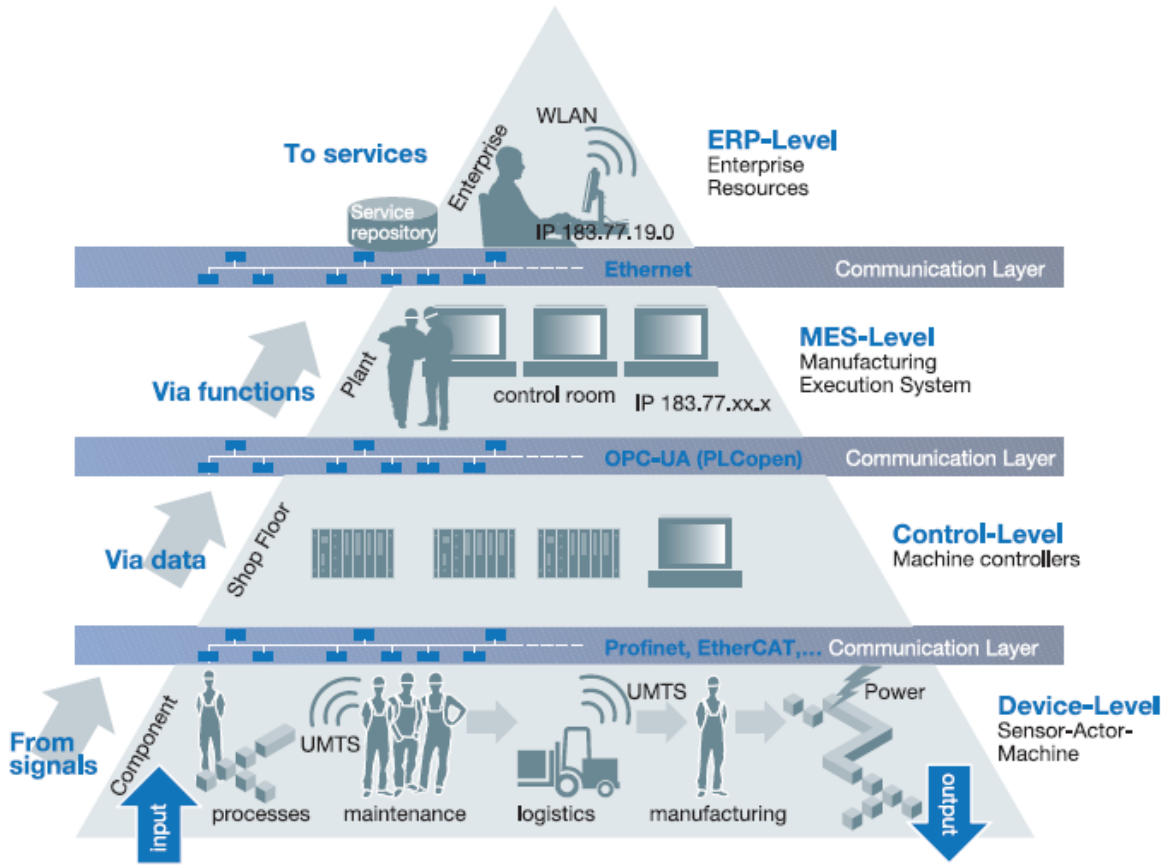
Joined Working Group PLCopen and OPCF

AGREEMENT OF USE

Contents

1. Scope

This specification was created by a joint working group of the OPC Foundation and PLCopen. It defines a set of *FUNCTION BLOCKS for OPC-UA client functionality (DA and method calls)*.
The interaction between IT and the world of automation is certainly not revolutionary, but corresponds with the established model of the automation pyramid:



This model is fundamentally based on the assumption that, in terms of communication, a controller as a main component of the automation system is “dumb”, and always merely responds to requests “from above”. The higher level is always the client and initiates data requests – the lower layer is always the server and courteously responds. In the modern world the strict separation of levels and the top-down approach of the information flow soften and mix. In a smart network, every device or service must be able to initiate independent communication with all other services.

This document is about OPC-UA client functionality out of the IEC61131-3 controller: A controller can exchange complex data structures horizontally with other controllers independently from fieldbus system or vertically with other devices using an OPC-UA server call in an MES/ERP system in order to collect data or write new production orders to the cloud. It allows a production line to be independently active in combination with integrated OPC UA Security features.

OPC-UA client functionality in a controller does not provide hard deterministic real time and so it’s not a deterministic fieldbus – but UA provides fast, secured communication providing modelling mechanism for information models.

Note: The FUNCTION BLOCKS are based on the second Edition of IEC61131-3.

OPC Foundation

The OPC Foundation defines standards for online data exchange between automation systems. They address access to current data (OPC DA), alarms and events (OPC A&E) and historical data (OPC HDA). Those standards are successfully applied in industrial automation.

The new OPC Unified Architecture (OPC-UA) unifies the existing standards and brings them to state-of-the-art technology using service-oriented architecture (SOA). Platform-independent technology allows the deployment of OPC-UA beyond current OPC applications only running on Windows-based PC systems. OPC-UA can also run on embedded systems as well as Linux / UNIX based enterprise systems. The provided information can be generically modelled and therefore arbitrary information models can be provided using OPC-UA.

PLCopen

PLCopen, as an organization active in industrial control, is creating a higher efficiency in your application software development: in one-off projects as well as in higher volume products. As such it is based on standard available tools to which extensions are and will be defined.

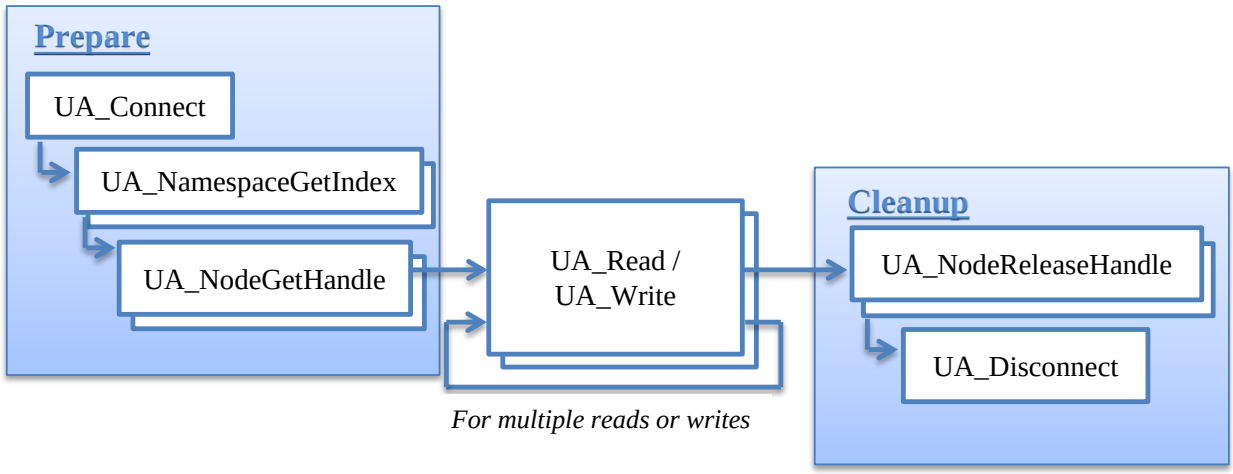
With results like Motion Control Library, Safety, XML specification, Reusability Level and Conformity Level, PLCopen made solid contributions to the community, extending the hardware independence from the software code, as well as reusability of the code and coupling to external software tools. One of the core activities of PLCopen is focused around IEC 61131-3, the only global standard for industrial control programming. It harmonizes the way people design and operate industrial controls by standardizing the programming interface. This allows people with different backgrounds and skills to create different elements of a program during different stages of the software lifecycle: specification, design, implementation, testing, installation and maintenance. Yet all pieces adhere to a common structure and work together harmoniously.

2. The Block Diagram

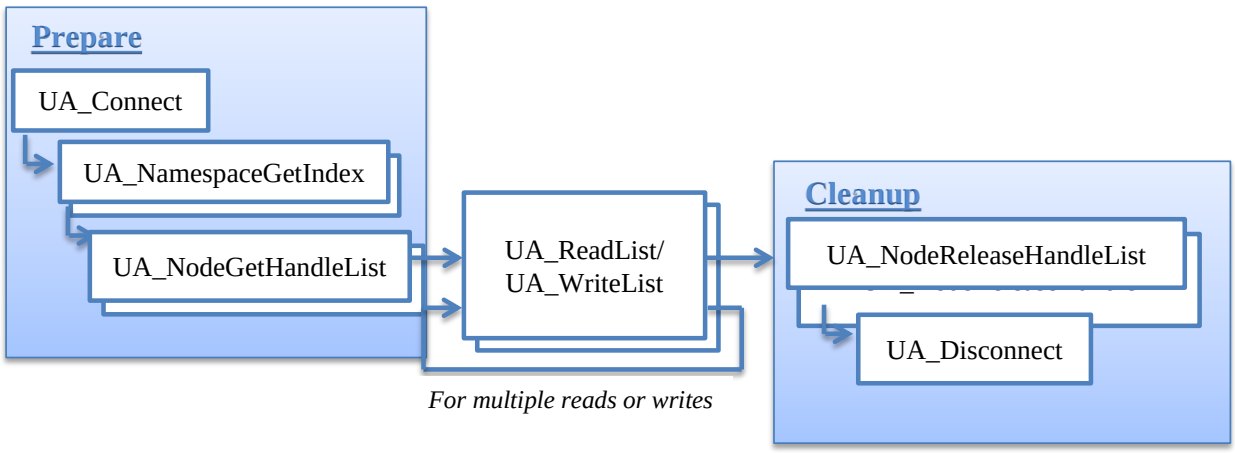
In order to perform an operation like UA_Read, UA_Write, UA_ReadList, UA_WriteList or UA_MethodCall following sequence of calls is required:

2.1. Read and Write

UA_Connect is to be performed once for each connection. The UA_NamespaceGetIndex is to be performed once for each namespace. The NodeHdl for a specific node is to be retrieved once. Read and write can be performed as frequent as necessary and permitted by the system. Once the purpose is served release the node handle not required anymore: use UA_ReleaseNodeHdl. Connection handle must be released using UA_Disconnect.



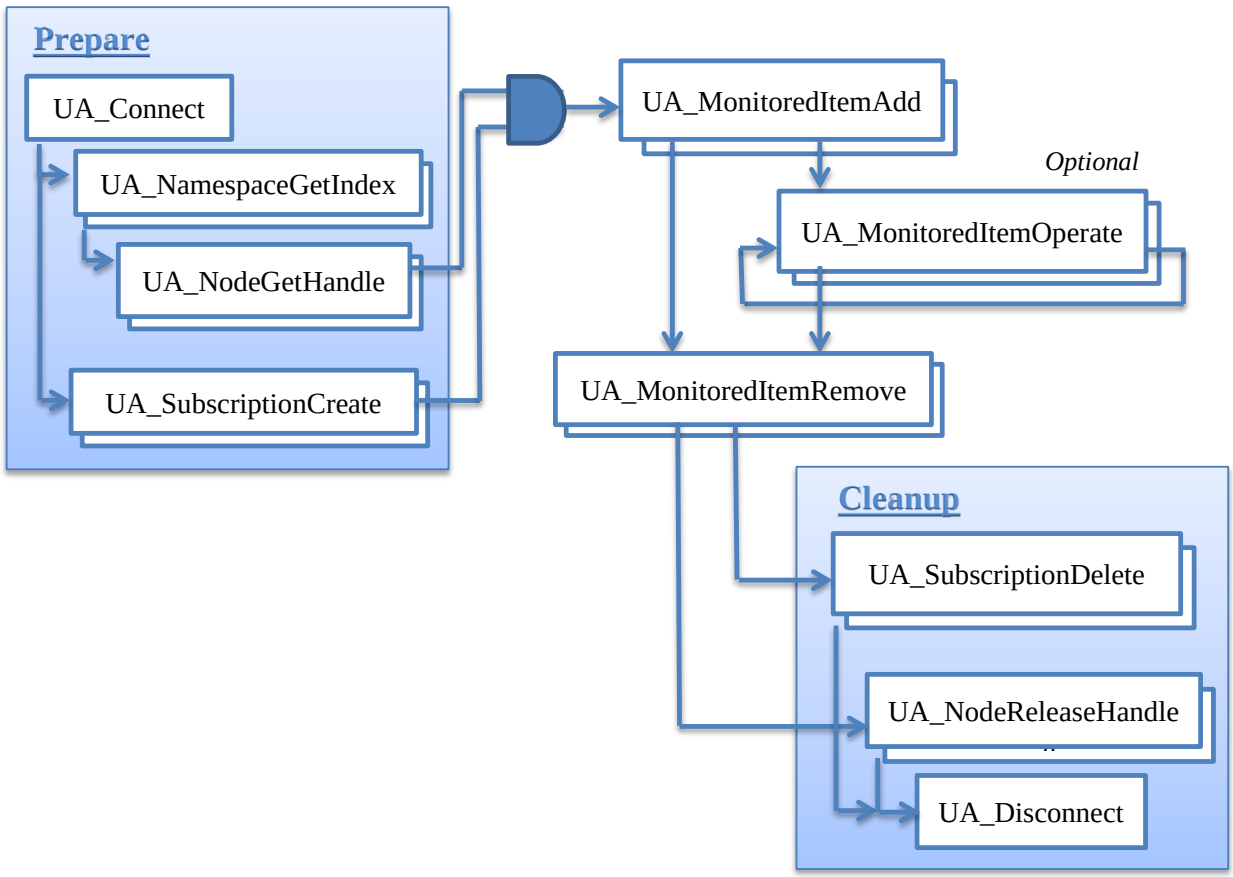
In addition to access the single elements there are function blocks which handle lists:



A list is handled as an array of the related base type (e.g. UANodeID or UANodeAdditionalInfo). Additionally there is a length which holds the number of elements in the array. Although several arrays can be connected to the function block (e.g. node handles and variables in case of UA_ReadList) there is only one length because all arrays have the same number of elements to be processed.
Please note that UA_NodeGetHandleList may not be able to resolve all input UANodeIDs. For such an unresolvable node the function block writes a value 0 (indicating an invalid handle) into the corresponding element of the output array. This output array can be used unchanged for subsequent calls to function blocks UA_ReadList, UA_WriteList and UA_NodeReleaseHandleList which do not perform any operation on nodeIds with the value 0.

2.2. Monitored Items

Following function blocks are to be used to create subscriptions and to add monitored items to this subscription.
To create a subscription, a valid connection handle is required. The connection handle is to be acquired using UA_Connect. UA_SubscriptionCreate will create a subscription. The SubscriptionHdl will be returned on successful execution of the function block UA_SubscriptionCreate. In order to monitor an item, NodeHdl for that specific node is required. In other words, both UA_SubscriptionCreate and UA_NodeGetHandle are to be called before calling UA_MonitoredItemAdd. UA_MonitoredItemAdd is used to add an item to a subscription mentioned by the SubscriptionHdl. The item to be monitored is to be assigned to this FB in form of NodeHdl. UA_MonitoredItemOperate can be used to modify monitoring settings like sampling interval, deadband type, deadband and hence is optional.
Take note to delete the subscription. Release the NodeHdl before you disconnect. If UA_NodeReleaseHandle is called before UA_SubscriptionDelete the Subscription will continue working.



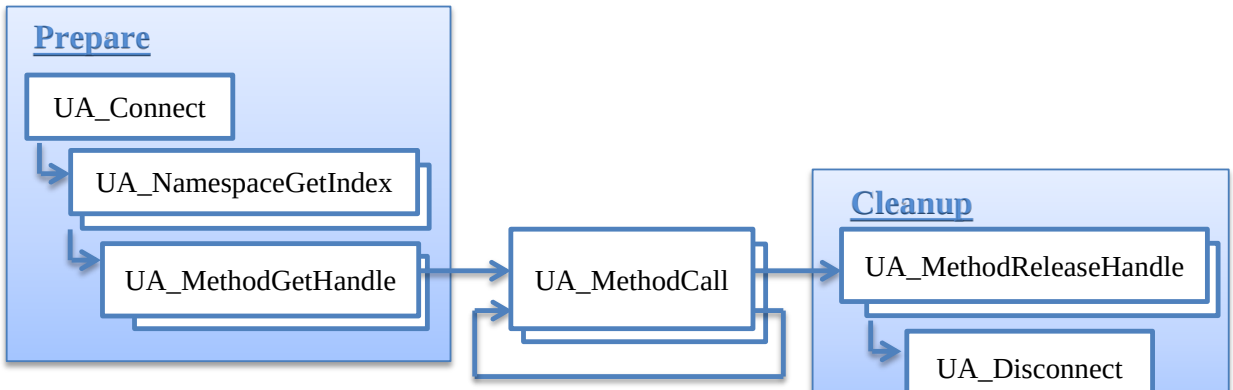
Monitoring of nodes does invert the communication interaction: The PLC is initiating the communication but as a consequence the values will be pushed from the UA-Server to the PLC.
Like shown in the block diagram above, a subscriptions and monitored items have to be set up.
There are two ways to actually retrieve latest values within the PLC:

- PLC-sync: Using the UA_SubscriptionOperate and UA_MonitoredItemOperate function blocks updated values could be retrieved.
  This means the PLC could decide when values are updated; but the function blocks need to be triggered.
- FW-sync: The firmware could internally update the values of the PLC memory.
  After adding monitored items the PLC does not need to care about the update procedure – and has no control the time when updates are available

The vendor has to decide which to provide. Vendor documentation should describe the selected behavior. Especially for the FW-sync way a detailed documentation should state when (i.e. Beginning of PLC cycle) updates are available to the PLC program.

2.3. Calling Methods

The appropriate sequence for calling methods is shown below. A valid method handle is necessary to call a method.
Successful call of UA_MethodGetHandle will deliver a valid MethodHdl.
Please release the method handle before you disconnect.



For multiple method calls

2.4. Diagnostics

Function block UA_Connect will deliver the ConnectionHdl. UA_ConnectionGetStatus requires this ConnectionHdl as input to deliver the connection status. In case the connection is lost after receiving the handle and while calling the UA_ConnectionGetStatus, ServerState Unknown will be returned.
NOTE: It is recommended to call UA_ConnectionGetStatus periodically but for performance reasons not in every PLC cycle.

3. Derived data types used in this specification

Within the specification the following derived data types are defined. Define which of these structures are used in this system:

Derived data types:	Where used	Supported	Which structure
UAUserIdentityToken	UASessionConnectInfo		
UASessionConnectInfo	UA_Connect		
UANodeID	UA_TranslatePath UA_NodeGetHandle UA_NodeGetInfo UA_MethodGetHandle		
UAMonitoringSettings	UA_NodeGetHandle		
UANodeInfo	UA_NodeGetInfo		
UAIndexRange	UA_MonitoredItemAdd UA_MonitoredItemRemove UA_MonitoredItemOperate UA_Read UA_ReadList UA_Write UA_WriteList		
UASecurityMsgMode	UA_Connect UA_SessionGetInfo		
UASecurityPolicy	UA_Connect		
UATransportProfile	UA_Connect		
UAUserIdentityTokenType	UA_Connect		
UAIdentifierType	UA_TranslatePath UA_NodeGetHandle UA_NodeGetInfo UA_MethodGetHandle		
UADeadbandType	UA_NodeGetHandle		
UANodeClass	UA_NodeGetInfo		
UAAttributeID	UA_MonitoredItemAdd UA_MonitoredItemRemove UA_MonitoredItemOperate UA_Read UA_ReadList UA_Write UA_WriteList		
UAConnectionStatus	UA_ConnectionGetStatus		
UAServerState	UA_ConnectionGetStatus		

Table 1: Supported derived data types

3.1. Enumerations

3.1.1. UASecurityMsgMode

Value	UASecurityMsgMode	Description
0	UASecurityMsgMode_BestAvailable	Best available message security mode to the UA server. The client receives the available message security from the server and selects the best. This could also result in level "none security".
1	UASecurityMsgMode_None	No security is applied.
2	UASecurityMsgMode_Sign	All messages are signed but not encrypted.
3	UASecurityMsgMode_SignEncrypt	All messages are signed and encrypted.

3.1.2. UASecurityPolicy

Value	UASecurityPolicy	Description
0	UASecurityPolicy_BestAvailable	Provides the best available security connection to the UA server. The client receives the available policies from the server and selects the best. This can also result in level "none security".
1	UASecurityPolicy_None	See OPC UA Part 7 Chapter 6.5.123
2	UASecurityPolicy_Basic128Rsa15	See OPC UA Part 7 Chapter 6.5.124
3	UASecurityPolicy_Basic256	See OPC UA Part 7 Chapter 6.5.125
4	UASecurityPolicy_Basic256Sha256	See OPC UA Part 7 Chapter 6.5.126

3.1.3. UATransportProfile

Value	UATransportProfile	Description
1	UATP_UATcp	See OPC UA Part 7 Chapter 6.5.107
2	UATP_WSHttpBinary	See OPC UA Part 7 Chapter 6.5.109
3	UATP_WSHttpXmlOrBinary	See OPC UA Part 7 Chapter 6.5.110
4	UATP_WSHttpXml	See OPC UA Part 7 Chapter 6.5.108

3.1.4. UAUserIdentityTokenType

Value	UAUserIdentityTokenType	Description
0	UAUITT_Anonymous	See OPC UA Part 7 Chapter 6.5.98
1	UAUITT_Username	See OPC UA Part 7 Chapter 6.5.99
2	UAUITT_x509	See OPC UA Part 7 Chapter 6.5.100
3	UAUITT_IssuedToken	See OPC UA Part 7 Chapter 6.5.101 (Not supported yet)

3.1.5. UAIdentifierType

Value	UAIdentifierType	Description
1	UAIdentifierType_String	see OPC UA Part 3 or Part 6
2	UAIdentifierType_Numeric	see OPC UA Part 3 or Part 6
3	UAIdentifierType_GUID	see OPC UA Part 3 or Part 6
4	UAIdentifierType_Opaque	see OPC UA Part 3 or Part 6

3.1.6. UADeadbandType

Value	UADeadbandType	Description
0	UADeadbandType_None	No Deadband calculation should be applied
1	UADeadbandType_Absolute	AbsoluteDeadband (See OPC UA Part 4, Chapter 7.16.2)
2	UADeadbandType_Percent	PercentDeadband (See OPC UA Part 4, Chapter 7.16.2)

3.1.7. UANodeClass

Value	UANodeClass	Description
0	UANodeClass_None	No node class (unspecified).
1	UANodeClass_Object	See OPC UA Part 3 Chapter 8.30

2	UANodeClass_Variable	See OPC UA Part 3 Chapter 8.30
4	UANodeClass_Method	See OPC UA Part 3 Chapter 8.30
8	UANodeClass_ObjectType	See OPC UA Part 3 Chapter 8.30
16	UANodeClass_VariableType	See OPC UA Part 3 Chapter 8.30
32	UANodeClass_ReferenceType	See OPC UA Part 3 Chapter 8.30
64	UANodeClass_DataType	See OPC UA Part 3 Chapter 8.30
128	UANodeClass_View	See OPC UA Part 3 Chapter 8.30
255	UANodeClass_All	All node classes combined.

3.1.8. UAAttributeID

Value	UAAttributeID	Description
1	UAAI_NodeID	The canonical identifier for the node.
2	UAAI_NodeClass	The class of the node.
3	UAAI_BrowseName	A non-localized, human readable name for the node.
4	UAAI_DisplayName	A localized, human readable name for the node.
5	UAAI_Description	A localized description for the node.
6	UAAI_WriteMask	Indicates which attributes are writeable.
7	UAAI_UserWriteMask	Indicates which attributes are writeable by the current user.
8	UAAI_IsAbstract	Indicates that a type node may not be instantiated.
9	UAAI_Symmetric	Indicates that forward and inverse references have the same meaning.
10	UAAI_InverseName	The browse name for an inverse reference.
11	UAAI_ContainsNoLoops	Indicates that following forward references within a view will not cause a loop.
12	UAAI_EventNotifier	Indicates that the node can be used to subscribe to events.
13	UAAI_Value	The value of a variable.
14	UAAI_DataType	The node id of the data type for the variable value.
15	UAAI_ValueRank	The number of dimensions in the value.
16	UAAI_ArrayDimensions	The length for each dimension of an array value.
17	UAAI_AccessLevel	How a variable may be accessed.
18	UAAI_UserAccessLevel	How a variable may be accessed after taking the user's access rights into account.
19	UAAI_MinimumSamplingInterval	Specifies (in milliseconds) how fast the server can reasonably sample the value for changes.
20	UAAI_Historizing	Specifies whether the server is actively collecting historical data for the variable.
21	UAAI_Executable	Whether the method can be called.
22	UAAI_UserExecutable	Whether the method can be called by the current user.

3.1.9. UAConnectionStatus

Value	UAConnectionStatus	Description
0	UACS_Connected	UA client is connected to UA server.
1	UACS_ConnectionError	The connection from UA client to UA server has an error.
2	UACS_Shutdown	The UA client has been disconnected from the UA server.

3.1.10. UAServerState

Value	UAServerState	Description
0	UASS_Running	The server is running normally. This is the usual state for a server.
1	UASS_Failed	A vendor-specific fatal error has occurred within the server. The server is no longer functioning. The recovery procedure from this situation is vendor-specific. Most *Service* requests should be expected to fail.
2	UASS_NoConfiguration	The server is running but has no configuration information loaded and therefore does not transfer data.

3	UASS_Suspended	The server has been temporarily suspended by some vendor-specific method and is not receiving or sending data.
4	UASS_Shutdown	The server has shut down or is in the process of shutting down. Depending on the implementation, this might or might not be visible to clients.
5	UASS_Test	The server is in Test Mode. The outputs are disconnected from the real hardware, but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. StatusCode will generally be returned normally.
6	UASS_CommunicationFault	The server is running properly, but is having difficulty accessing data from its data sources. This may be due to communication problems or some other problems preventing the underlying device, control system, etc. from returning valid data. It may be a complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD FAILURE status code indication for the items.
7	UASS _Unknown	This state is used only to indicate that the OPC UA server does not know the state of underlying servers.

3.2. Structure

3.2.1. UAUserIdentityToken

UAUserIdentityToken	DataType	Description
UserIdentityTokenType	UAUserIdentityTokenType	Defines the identity Token to authenticate a user during the creation of a Session. See 3.1.4 UAUserIdentityTokenType.
TokenParam1	STRING	In case of TokenType “Anonymous” the Param1 will not be evaluated. In case of TokenType “Username” the Param1 contains the user name. In case of TokenType “x509” the Param1 contains the location of the certificate store.
TokenParam2	STRING	In case of TokenType “Anonymous” the Param2 will not be evaluated. In case of TokenType “Username” the Param2 contains the user password. In case of TokenType “x509” the Param2 contains the certificate name.

3.2.2. UASessionConnectInfo

UASessionConnectInfo	DataType	Description
SessionName	STRING	Defines the name of the session assigned by the client. The name is shown in the diagnostics information of the server. In case of empty string the server will generate a session name.
ApplicationName	STRING	Defines the readable name of the OPC UA client application. The string can be empty.
SecurityMsgMode	UASecurityMsgMode	See 3.1.1 UASecurityMsgMode.
SecurityPolicy	UASecurityPolicy	See 3.1.2 UASecurityPolicy.
CertificateStore	STRING	Defines the location of the certificate store used for the application certificates and trust lists. The structure of the certificate store is vendor specific. In case of empty string the default certificate store is used.
ClientCertificateName	STRING	Defines the name of the client certificate and private key in the certificate store. In case of empty string the default client application certificate is used. Implementation note: The ApplicationURI will be extracted from the certificate.
ServerUri	STRING	Defines the URI of the server.
CheckServerCertificate	BOOL	Flag indicating if the server certificate should be checked with the trust list of the client application.
TransportProfile	UATransportProfile	See 3.1.3 UATransportProfile
UserIdentityToken	UAUserIdentityToken	See 3.2.1 UAUserIdentityToken
VendorSpecificParameter	STRING	Vendor may define specific parameters. e.g. In case multiple clients are available, client instance can be defined with this parameter. The string can be empty.
SessionTimeout	TIME	Defines how long the session will survive when there is no connection.
MonitorConnection	TIME	Defines the interval time to check the connection. The connection monitoring has to be done by the client vendor implementation and is defined by the OPC UA specification part 4.
LocaleIDs	ARRAY [1..5] OF STRING[6]	OPC-UA Part3 / Chapter 8.4: <language>[-<country/region>] where <language> is a two letter ISO639 code for language, <country /region> is

		the three letter ISO3166 code for the country/region. Sample: en-US, zh-CHS

3.2.3. UANodeID

UANodeID	DataType	Description
NamespaceIndex	UINT	
Identifier	STRING	In case of IdentifierType GUID the format is like 00000316-0000-0000-C000-000001000046 In case of IdentifierType Opaque string has to be base 64 encoded byte string.
IdentifierType	UAIdentifierType	See 3.1.5 UAIdentifierType

3.2.4. UAMonitoringSettings

UAMonitoringSettings	DataType	Description
SamplingInterval	TIME	The rate in milliseconds the server checks the underlying data source for changes.
DeadbandType	UADeadbandType	See 3.1.6 UADeadbandType. This parameter indicates if a deadband is applied and if applied, which type of Deadband.
Deadband	REAL	e.g. percent 0.1%

NOTE: The QueueSize for the monitoring items is set to 1. See OPA UA Specification Part 4.

3.2.5. UALocalizedText

UALocalizedText	DataType	Description
Locale	STRING[6]	OPC-UA Part3 / Chapter 8.4: <language>[-<country/region>] where <language> is a two letter ISO639 code for language, <country /region> is the three letter ISO3166 code for the country/region. Sample: en-US, zh-CHS
Text	STRING	Contains localized text as a string.

3.2.6. UANodeInfo

<table>
<tr><th>UANodeInfo</th><th>DataType</th><th>Description</th></tr>
<tr><td>AccessLevel</td><td>BYTE</td><td>A bit mask indicating whether the current value of the Value Attribute is readable and writable as well as whether the history of the value is readable and changeable.
<table>
<tr><th>Bit</th><th>Value</th><th>AccessLevel</th></tr>
<tr><td></td><td>0x0</td><td>None</td></tr>
<tr><td>0</td><td>0x1</td><td>CurrentRead</td></tr>
<tr><td>1</td><td>0x2</td><td>CurrentWrite</td></tr>
<tr><td>2</td><td>0x4</td><td>HistoryRead</td></tr>
<tr><td>3</td><td>0x8</td><td>HistoryWrite</td></tr>
<tr><td>4</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>ArrayDimension</td><td>ARRAY OF UDINT</td><td>The length for each dimension of an array value.</td></tr>
<tr><td>BrowseName</td><td>STRING</td><td>The BrowseName is composed of a namespace index and a name. The String representation is of the format [ns:]BrowseName. The browse name may be prefixed by its namespace index. If the namespace prefix is omitted then namespace index 0 is used.</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>ContainsNoLoops</td><td>BOOL</td><td>Indicates that following forward references within a view will not cause a loop.</td></tr>
<tr><td>DataType</td><td>UANodeID</td><td>See 3.2.3 UANodeID. The node id of the data type for the variable value.</td></tr>
<tr><td>Description</td><td>UALocalizedText</td><td>A localized description for the node.</td></tr>
<tr><td>DisplayName</td><td>UALocalizedText</td><td>A localized human readable name for the node.</td></tr>
<tr><td>EventNotifier</td><td>BYTE</td><td>This Attribute represents a bit mask that identifies whether the Object can be used to subscribe to Events and whether the history of Events is accessible and changeable.
<table>
<tr><th>Bit</th><th>Value</th><th>EventNotifier</th></tr>
<tr><td></td><td>0x0</td><td>The Object does not produce events and has no event history. (SubscribeToEvents)</td></tr>
<tr><td>0</td><td>0x1</td><td>The Object produces event notifications.</td></tr>
<tr><td>1</td><td>0x2</td><td>Reserved. Must always be zero</td></tr>
<tr><td>2</td><td>0x4</td><td>The Object has an event history which may be read. (HistoryRead)</td></tr>
<tr><td>3</td><td>0x8</td><td>The Object has an event history which may be updated (HistoryWrite)</td></tr>
<tr><td>4</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Executable</td><td>BOOL</td><td>Whether the method can be called.</td></tr>
<tr><td>Historizing</td><td>BOOL</td><td>Specifies whether the server is actively collecting historical data for the variable.</td></tr>
<tr><td>InverseName</td><td>STRING</td><td>The browse name for an inverse reference.</td></tr>
<tr><td>IsAbstract</td><td>BOOL</td><td>Indicates that a type node may not be instantiated.</td></tr>
<tr><td>MinimumSamplingInterval</td><td>TIME</td><td>Specifies (in ms) how fast the server can reasonably sample the value for changes.</td></tr>
<tr><td>NodeClass</td><td>UANodeClass</td><td>See 3.1.7 UANodeClass. The base type of the node. An enumeration identifying the NodeClass of a Node such as Object, Variable or Method.</td></tr>
<tr><td>NodeID</td><td>UANodeID</td><td>See 3.2.3 UANodeID. The server unique identifier for the node.</td></tr>
<tr><td>Symmetric</td><td>BOOL</td><td>Indicates that forward and inverse references have the same meaning.</td></tr>
<tr><td>UserAccessLevel</td><td>BYTE</td><td>Contains the same information as the AccessLevel but takes user access rights into account.
<table>
<tr><th>Bit</th><th>Value</th><th>UserAccessLevel</th></tr>
<tr><td>0</td><td>0x1</td><td></td></tr>
<tr><td>1</td><td>0x2</td><td></td></tr>
<tr><td>2</td><td>0x4</td><td></td></tr>
<tr><td>3</td><td>0x8</td><td></td></tr>
<tr><td>4</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>UserExecutable</td><td>BOOL</td><td>Whether the method can be called by the current user.</td></tr>
<tr><td>UserWriteMask</td><td>UDINT</td><td>Indicates which attributes are writeable by the current user.</td></tr>
<tr><td>ValueRank</td><td>DINT</td><td>The number of dimensions in the value.</td></tr>
<tr><td>WriteMask</td><td>UDINT</td><td>Indicates which attributes are writeable.</td></tr>
</table>

3.2.7. UAIndexRange

UAIndexRange	DataType	Description
StartIndex	UINT	Start index.
EndIndex	UINT	End index.

Note: IndexRange can be defined as follows:
For each Dimension:
1. Start and EndIndex are to be assigned
2. StartIndex must be smaller than EndIndex
3. To access all the elements in a Dimension **it's a must to assign** StartIndex and EndIndex depending on the number of total Elements in the Dimension.
4. A single element in a Dimension can be selected by specifying the same StartIndex and EndIndex.

3.2.8. UANodeAdditionalInfo

UANodeAdditionalInfo	DataType	Description
AttributeID	UAAttributeID	Selects the attribute to be accessed. The default AttributeID is eUAAI_Value (13). See 3.1.8 UAAttributeID
IndexRangeCount	UINT	Count of valid IndexRange specified. Vendorspecific.
IndexRange	ARRAY OF UAIndexRange	See 3.2.7 UAIndexRange

3.3. Constants of Array Lengths

The described function blocks make use of arrays.
The length of these arrays is – if not formally limited by the function block –vendor-specific and could be made changeable for resource optimization.
This is a list of arrays and their length-constants as a naming convention.
Every group of arrays should have the same length for ease of use.
All arrays should be defined as [1..CONSTANT-LENGTH]

CONSTANT-LENGTH	Description
MAX_ELEMENTS_ ARRAYDIMENSION	Used at 3.2.6 *UANodeInfo*. Limits the maximum dimensions of a node, which could be used.
MAX_ELEMENTS_ INDEXRANGE	Used at 3.2.8 *UANodeAdditionalInfo* Limits the maximum defined. Could be equal to MAX_ELEMENTS_ ARRAYDIMENSION as a general dimension limit.
MAX_ELEMENTS_ NODELIST	Used at all List function blocks: 5.6 UA_NodeGetHandleList(for NodeIds, NodeHdls and NodeErrorIDs) 5.8 UA_NodeReleaseHandleList(for NodeHdls and NodeErrorIDs) 5.17 UA_ReadList(for NodeHdls, NodeAddInfos, NodeErrorIDs, Timestamps, Variables) 5.19 UA_WriteList(for NodeHdls, NodeAddInfos, NodeErrorIDs, Variables) Limits the number of nodes, which could be used by the List function blocks

4. Error Codes (ErrorID)

Error codes are 4 bytes long, data type being DWORD.
NOTE: Bit 29 in this DWORD value is used to differentiate between error codes defined by OPC Foundation and error codes defined by PLCopen or Vendor.

Field	Bit Range	Description
Severity	30:31	Indicates whether the *ErrorCode* represents a good, bad or uncertain condition. These bits have the following meanings: Good success — 00 — Indicates that the operation was successful and the associated results may be used. Uncertain Warning — 01 — Indicates that the operation was partially successful and that associated results might not be suitable for some purposes. Bad Failure — 10 — Indicates that the operation failed and any associated results cannot be used. Reserved — 11 — Reserved for future use. All *Clients* should treat an *ErrorCode* with this severity as "Bad".
ErrorType	29	Value 0 indicates OPC error. Please find these error codes in OPC UA specification. Value 1 indicates PLCopen or vendor specific error signaled by BIT28.
ErrorType2	28	This flag can be evaluated only if BIT29 is Value 1. Value 0 indicates PLCopen error. Value 1 indicates vendor specific error.

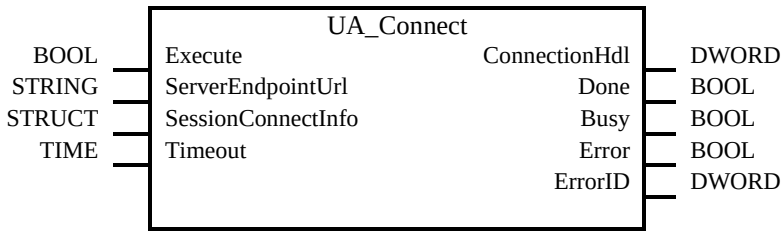
ErrorCode (Bits 0..27)	Define	Description
Category	**General**	
	PLCopenUA_Bad_FW_PermanentError	Internal, permanent error.
	PLCopenUA_Bad_FW_TempError	Temp. error; FB could retry to reach FW.
Category	**Connection**	
	PLCopenUA_Bad_ConnectionError	Connection could not be established.
	PLCopenUA_Bad_HostNotFound	The requested hostname could not be found.
	PLCopenUA_Bad_AlreadyConnected	Connection was already established.
	PLCopenUA_Bad_SecurityFailed	Connection failed due to security setup.
	PLCopenUA_Bad_Suspended	Connection is suspended.
	PLCopenUA_Bad_ConnectionInvalidHdl	Provided ConnectionHdl is not known.
Category	**Namespace**	
	PLCopenUA_Bad_NSNotFound	A namespace with the requested name cannot be found on server.
Category	**Node**	
	PLCopenUA_Bad_ResultTooLong	Target PLC variable is too short for retrieved data.
	PLCopenUA_Bad_InvalidType	Invalid or unsupported Type.
	PLCopenUA_Bad_NodeInvalidHdl	Provided NodeHdl is not known.
	PLCopenUA_Bad_MethodInvalidHdl	Provided MethodHdl is not known.
	PLCopenUA_Bad_ReadFailed	Read failed for unknown reason.

	PLCopenUA_Bad_WriteFailed	Write failed for unknown reason.
	PLCopenUA_Bad_CallFailed	Method Call failed for unknown reason.
	PLCopenUA_Bad_InParamFailed	Method Call Input parameter conversion failed .
	PLCopenUA_Bad_OutParamFailed	Method Call Output parameter conversion failed. ATTENTION: this means the MethodCall was executed successfully but the returned values could not be converted.
Category	**Monitoring**	
	PLCopenUA_Bad_SubscriptionInvalidHdl	Provided SubscriptionHdl is not known.
	PLCopenUA_Bad_MonitoredItemInvalidHdl	Provided MonitoredItemHdl is not known.

5. Data Communication

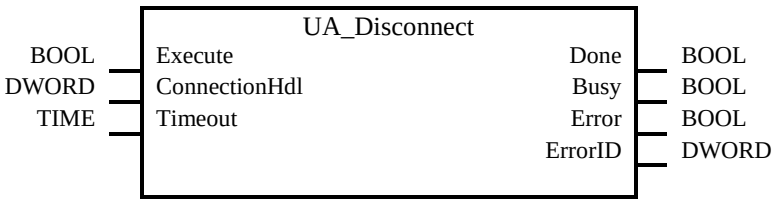
5.1. UA_Connect

FB-Name		**UA_Connect**	
This Function Block is used to create a (optional secure) transport connection and an OPC-UA session. The connection shall be terminated by calling the UA_Disconnect after establishing the connection.			
VAR_INPUT			
	Execute	BOOL	On rising edge connection is started.
	ServerEndpointUrl	STRING	URL
	SessionConnectInfo	STRUCT	See 3.2.2 UASessionConnectInfo
	Timeout	TIME	Maximum time to establish the connection.
VAR_OUTPUT			
	ConnectionHdl	DWORD	Connection handle – is valid until UA_Disconnect is called.
	Done	BOOL	Signals a connection has been initially established.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: The connection monitoring and the reconnect handling are to be done by the client vendor implementation. The reconnect sequence is defined by the OPC UA specification part 4.			



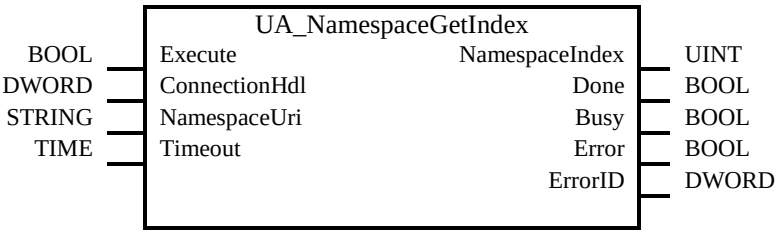
5.2. UA_Disconnect

FB-Name		**UA_Disconnect**	
This Function Block is used to close a transport connection and an OPC-UA session.			
VAR_INPUT			
	Execute	BOOL	On rising edge connection is terminated.
	ConnectionHdl	DWORD	Connection handle of connection to be closed.
	Timeout	TIME	Maximum time to close the connection.
VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: Calling UA_Disconnect (even in case of timeout or error) will release the ConnectionHdl, all node-handles and MonitoredItems.			



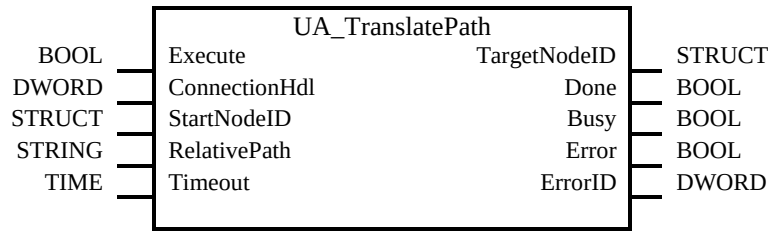
5.3. UA_NamespaceGetIndex

FB-Name	**UA_NamespaceGetIndex**	
This Function Block is used to get the namespace-index of a namespace-URI		
VAR_INPUT		
Execute	BOOL	FB performs its task on rising edge on this input.
ConnectionHdl	DWORD	Connection handle.
NamespaceUri	STRING	Namespace URI.
Timeout	TIME	Maximum time to response.
VAR_OUTPUT		
NamespaceIndex	UINT	Namespace Index.
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
Notes: -		



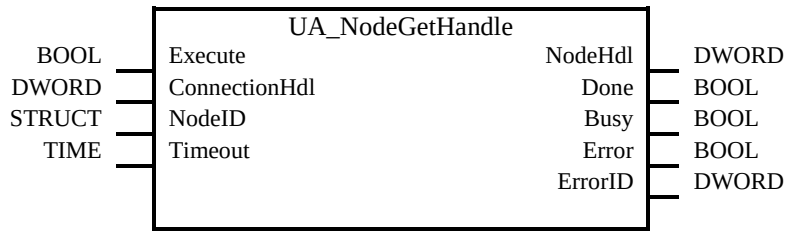
5.4. UA_TranslatePath

FB-Name	**UA_TranslatePath**	
This Function Block is used to get the node parameters of a node using path of the node.		
VAR_INPUT		
Execute	BOOL	FB performs its task on rising edge on this input.
ConnectionHdl	DWORD	Connection handle.
StartNodeID	STRUCT	See 3.2.3 UANodeID. Structure UANodeID with node parameters for starting node.
RelativePath	STRING	Path of the Target node; BNF of RelativePath is defined in the OPC UA specification Part 4.
Timeout	TIME	Time to response.
VAR_OUTPUT		
TargetNodeID	STRUCT	See 3.2.3 UANodeID. Structure UANodeID with node parameters. For target node mentioned by RelativePath at the input of this FB.
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
Notes: -		



5.5. *UA_NodeGetHandle*

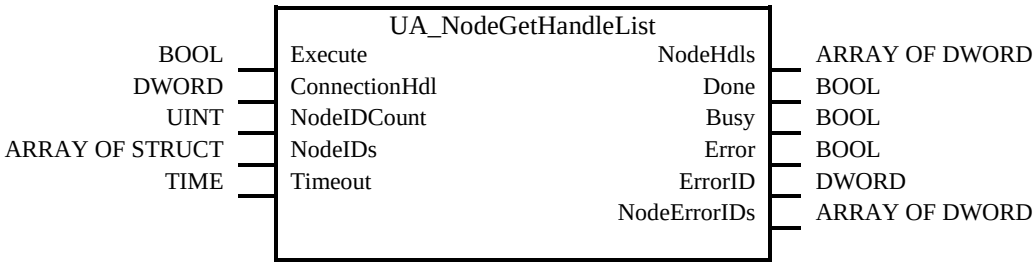
FB-Name	**UA_NodeGetHandle**		
This Function Block is used to get the node handle.			
VAR_INPUT			
	Execute	BOOL	FB performs its task on rising edge on this input.
	ConnectionHdl	DWORD	Connection handle.
	NodeID	STRUCT	See 3.2.3 UANodeID
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	NodeHdl	DWORD	Node handle.
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: The NodeHdl is a reference to the internal management object for the node in the client. But the client shall also register the node at the server ("RegisterNode"). This enables the UA-server to optimize the communication. The scope of the NodeHdl is the connection. So a NodeHdl is unique for a connection but could be equal to a NodeHdl of another connection.			



5.6. *UA_NodeGetHandleList*

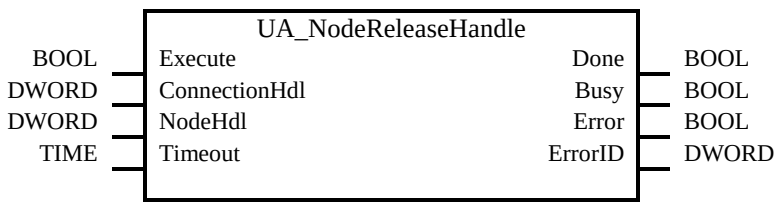
FB-Name	**UA_NodeGetHandleList**		
This Function Block is used to get node handles for multiple nodes.			
VAR_INPUT			
	Execute	BOOL	FB performs its task on rising edge on this input.
	ConnectionHdl	DWORD	Connection handle.
	NodeIDCount	UINT	Number of NodeIDs in Array of NodeIDs.
	NodeIDs	ARRAY OF STRUCT	See 3.2.3 *UANodeID*. Max length of array is to be defined by the vendor. Array length of NodeIDs and NodeHdls must be same.
	Timeout	TIME	Time to response.
VAR_OUTPUT			

	NodeHdls	ARRAY OF DWORD	Array of Node Handles. Max length of array is to be defined by the vendor. Array length of NodeIDs and NodeHdls must be same.
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB. Set to true if either ErrorID or any of the NodeErrorIDs indicates an error.
	ErrorID	DWORD	Error code.
	NodeErrorIDs	ARRAY OF DWORD	Array of NodeErrorIDs. Contains an error code for each valid element of the NodeIds array. Max length of array is to be defined by the vendor and shall be same size like the NodesIDs array length.
Notes: The NodeHdl is a reference to the internal management object for the node in the client. But the client shall also register the node at the server ("RegisterNode"). This enables the UA-server to optimize the communication.			



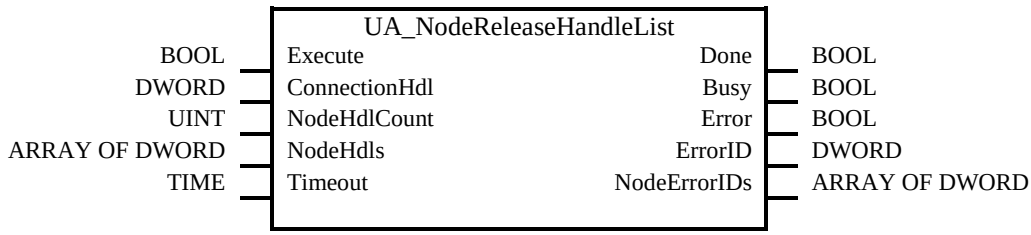
5.7. UA_NodeReleaseHandle

FB-Name	**UA_NodeReleaseHandle**	
This Function Block is used to release the node handle.		
VAR_INPUT		
Execute	BOOL	FB performs its task on rising edge on this input.
ConnectionHdl	DWORD	Connection handle.
NodeHdl	DWORD	Node handle to be released .
Timeout	TIME	Time to response.
VAR_OUTPUT		
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
Notes: After calling UA_NodeReleaseHandle the NodeHdl will be invalid.		



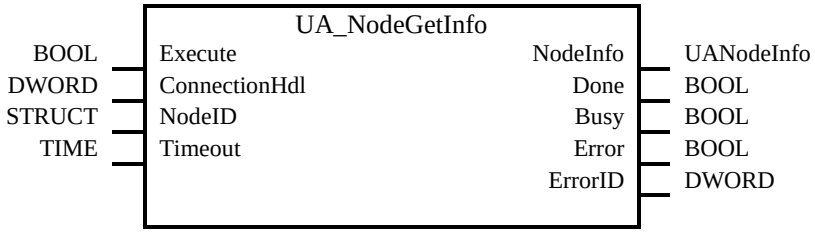
5.8. UA_NodeReleaseHandleList

FB-Name		**UA_NodeReleaseHandleList**	
This Function Block is used to release a set of node handles.			
VAR_INPUT			
	Execute	BOOL	FB performs its task on rising edge on this input.
	ConnectionHdl	DWORD	Connection handle.
	NodeHdlCount	UINT	Number of Nodes in NodeHdls Array.
	NodeHdls	ARRAY OF DWORD	Array of Node handles to be released. Max length of array is to be defined by the vendor. NULL is not a valid handle.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB. Set to true if either ErrorID or any of the NodeErrorIDs indicates an error.
	ErrorID	DWORD	Error code.
	NodeErrorIDs	ARRAY OF DWORD	Array of DWORD. Contains an error code for each valid element of the NodeHdls array. Max length of array is to be defined by the vendor and shall be same size like the NodeHdls array length.
Notes: After calling UA_NodeReleaseHandleList the NodeHdls will be invalid.			



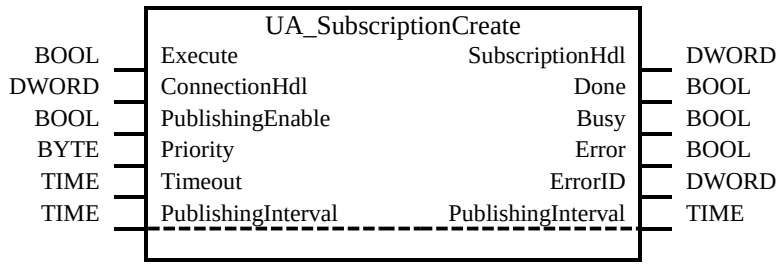
5.9. UA_NodeGetInfo

FB-Name		**UA_NodeGetInfo**	
This Function Block is used to get the node information.			
VAR_INPUT			
	Execute	BOOL	On rising edge node information will be read.
	ConnectionHdl	DWORD	Connection handle.
	NodeID	STRUCT	See 3.2.3 UANodeID
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	NodeInfo	STRUCT	See 3.2.6 UANodeInfo
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: -			



5.10. UA_SubscriptionCreate

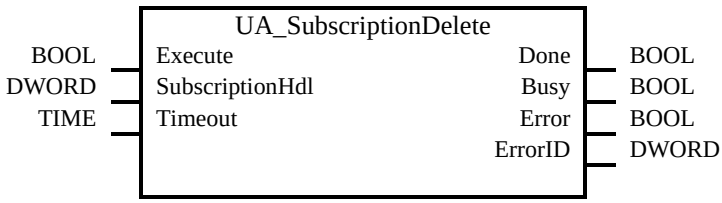
FB-Name	**UA_SubscriptionCreate**	
This Function Block can be used to create a subscription.		
VAR_INPUT		
Execute	BOOL	On rising edge subscription will be created.
ConnectionHdl	DWORD	Connection handle.
PublishingEnable	BOOL	Activate the publishing.
Priority	BYTE	Priority of the Subscription in the server relative to the other Subscriptions created by this client.
Timeout	TIME	Maximum time to response.
VAR_OUTPUT		
SubscriptionHdl	DWORD	Subscription handle.
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
VAR_IN_OUT		
PublishingInterval	TIME	Publishing interval (can be changed by the Server revised publishing interval).
Notes: The connection monitoring and the reconnect handling are to be done by the client vendor implementation. The reconnect sequence is defined by the OPC UA specification part 4. SubscriptionHdl must be unique even if the client is connected to multiple servers.		



5.11. UA_SubscriptionDelete

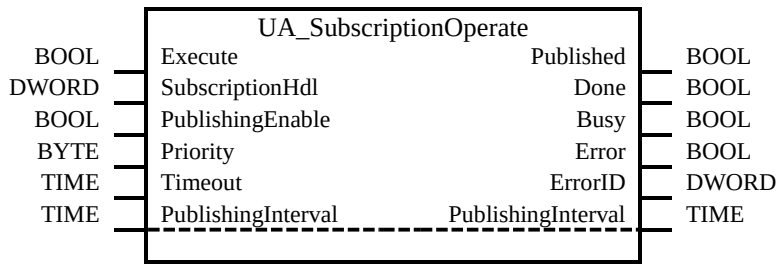
FB-Name	**UA_SubscriptionDelete**	
This Function Block can be used to delete a subscription.		
VAR_INPUT		
Execute	BOOL	On rising edge the subscription mentioned by SubscriptionHdl will be deleted.
SubscriptionHdl	DWORD	Subscription handle.
Timeout	TIME	Time to response.

VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: -			



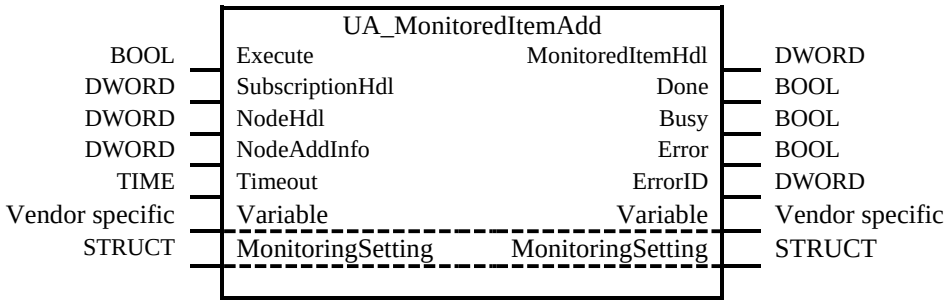
5.12. UA_SubscriptionOperate

FB-Name	**UA_SubscriptionOperate**	
This Function Block is designed to be optionally called -even cyclically- to check if the variables have been published and to check and modify publishing parameters (enable / interval).		
VAR_INPUT		
Execute	BOOL	FB operates on rising edge.
SubscriptionHdl	DWORD	Subscription handle.
PublishingEnable	BOOL	Activates the publishing.
Priority	BYTE	Priority of the Subscription in the server relative to the other Subscriptions created by this client.
Timeout	TIME	Time to response.
VAR_OUTPUT		
Published	BOOL	Indicates, that variables have been published since the previous call.
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
VAR_IN_OUT		
PublishingInterval	TIME	Publishing interval (can be changed by the Server revised publishing interval).
Notes: -		



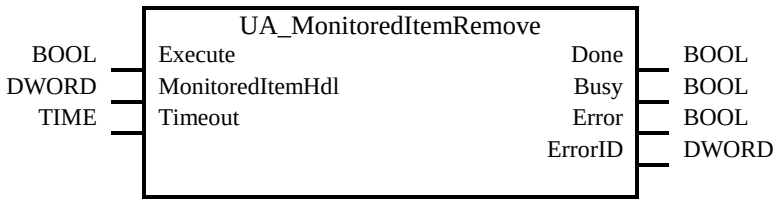
5.13. UA_MonitoredItemAdd

FB-Name	**UA_MonitoredItemAdd**		
This Function Block can be used to add handle that values are updated by subscription.			
VAR_INPUT			
	Execute	BOOL	On rising edge monitored item will be added to a subscription.
	SubscriptionHdl	DWORD	Subscription handle.
	NodeHdl	DWORD	Node handle.
	NodeAddInfo	DWORD	See 3.2.8 UANodeAdditionalInfo. Specifies the attribute and IndexRange.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	MonitoredItemHdl	DWORD	Monitored item handle.
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
VAR_IN_OUT			
	Variable	Vendor specific	To be defined by vendor.
	MonitoringSettings	STRUCT	See 3.2.4 UAMonitoringSettings
Notes: VAR_IN_OUT: „Variable" as would provide best type save solution for users: The client firmware is able to map the UA memory layout to the controller layout. The firmware client can receive the type definition from the UA-Server. Workaround would be to provide a byte array as "Variable" and the firmware client just provide the blob (UA memory layout – so called "raw data") into that byte array. "Variable" could be the name of the variable so the internal firmware can get address, length, data type of variable.			



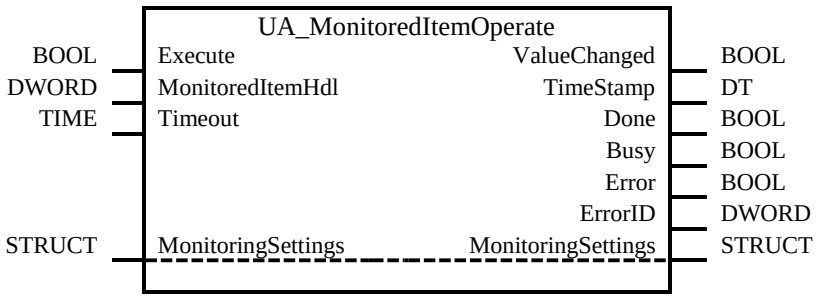
5.14. UA_MonitoredItemRemove

FB-Name	**UA_MonitoredItemRemove**		
This Function Block can be used to remove a handle from a subscription.			
VAR_INPUT			
	Execute	BOOL	On rising edge node information will be read.
	MonitoredItemHdl	DWORD	Monitored item handle.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: -			



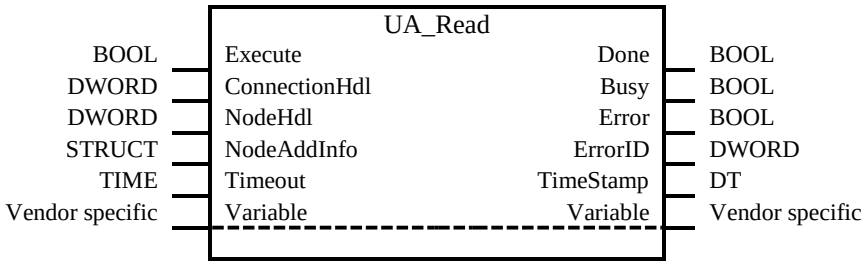
5.15. UA_MonitoredItemOperate

FB-Name	**UA_MonitoredItemOperate**		
This Function Block is designed to be optionally called to check and modify monitored item parameters.			
VAR_INPUT			
	Execute	BOOL	On rising edge node information will be read.
	MonitoredItemHdl	DWORD	Monitored item handle.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	ValueChanged	BOOL	Indicates that the value of the monitored item has been changed.
	TimeStamp	DT	TimeStamp
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
VAR_IN_OUT			
	MonitoringSettings	STRUCT	See 3.2.4 UAMonitoringSettings
Notes: -			



5.16. UA_Read

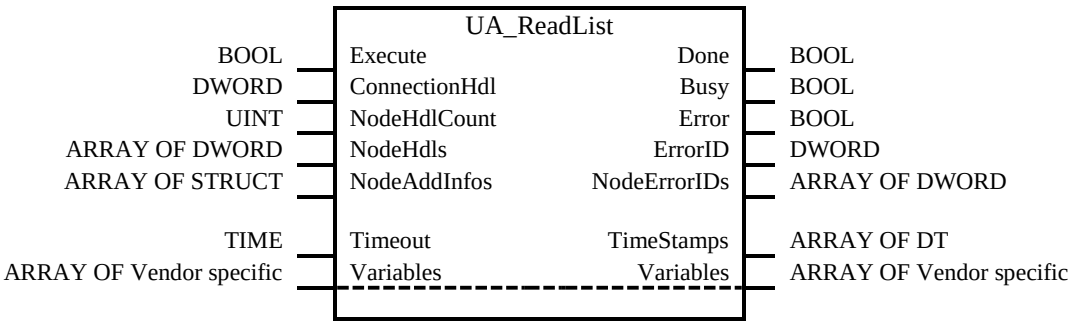
FB-Name	**UA_Read**		
This Function Block is used to read the value of a single node.			
VAR_INPUT			
	Execute	BOOL	On rising edge node information will be read.
	ConnectionHdl	DWORD	Connection handle.
	NodeHdl	DWORD	Node handle.
	NodeAddInfo	STRUCT	See 3.2.8 UANodeAdditionalInfo. Specifies the attribute and IndexRange.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
	TimeStamp	DT	TimeStamp.
VAR_IN_OUT			
	Variable	Vendor specific	Vendor specific (DATE_AND_TIME)
Notes: Vendors can handle "Variable" in a vendor specific way. Independent of the vendor specific solution the mapping of the controller data type and OPC-UA data type shall be handled in the function block. VAR_IN_OUT: „Variable" as would provide best type save solution for users: The client firmware is able to map the UA memory layout to the controller layout. The firmware client can receive the type definition from the UA-Server. Workaround would be to provide a byte array as "Variable" and the firmware client just provide the blob (UA memory layout – so called "raw data") into that byte array. "Variable" could be the name of the variable so the internal firmware can get address, length, data type of variable.			



5.17. UA_ReadList

FB-Name	**UA_ReadList**		
This Function Block is used to read values of multiple nodes using a list of node handles.			
VAR_INPUT			
	Execute	BOOL	On rising edge node information will be read.
	ConnectionHdl	DWORD	Connection handle.
	NodeHdlCount	UINT	Number of valid elements in the array to read.
	NodeHdls	ARRAY OF DWORD	Array of Node Handles. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
	NodeAddInfos	ARRAY OF STRUCT	See 3.2.8 UANodeAdditionalInfo. Array of UANodeAdditionalInfo. Specifies the attribute and IndexRange. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
	Timeout	TIME	Time to response.

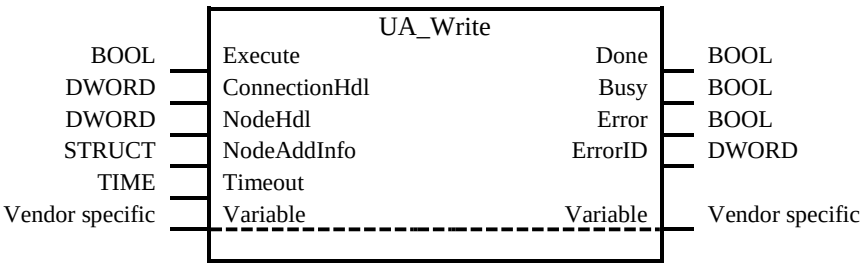
VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected
	Error	BOOL	Signals that an error has occurred within the FB. Set to true if either ErrorID or any of the NodeErrorIDs indicates an error.
	ErrorID	DWORD	Error code for the OPC UA service call.
	NodeErrorIDs	ARRAY OF DWORD	Array of DWORD. Contains an error code for each valid element of the Variables array. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
	TimeStamps	ARRAY OF DT	Contains a TimeStamp for each valid element of the Variables array. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
VAR_IN_OUT			
	Variables	ARRAY OF Vendor specific	Vendor specific. Max length of array is to be defined by the vendor.
Notes: Vendors can handle "Variable" in a vendor specific way. Independent of the vendor specific solution the mapping of the controller data type and OPC-UA data type shall be handled in the function block. VAR_IN_OUT: „Variable" as would provide best type save solution for users: The client firmware is able to map the UA memory layout to the controller layout. The firmware client can receive the type definition from the UA-Server. Workaround would be to provide a byte array as "Variable" and the firmware client just provide the blob (UA memory layout – so called "raw data") into that byte array. "Variable" could be the name of the variable so the internal firmware can get address, length, data type of variable.			



5.18. UA_Write

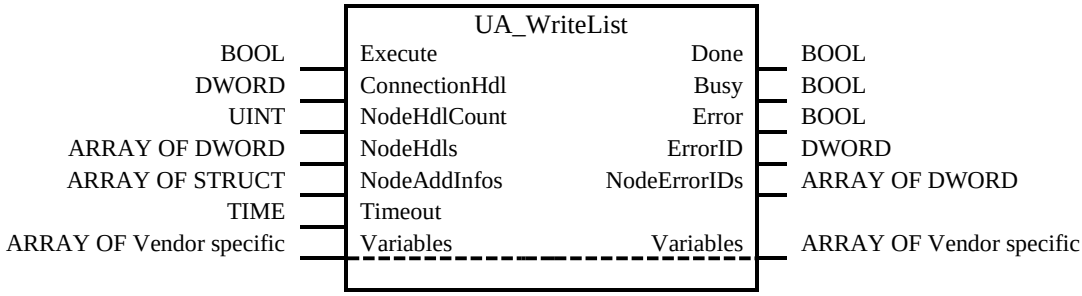
FB-Name	**UA_Write**		
This Function Block is used to write a value to a single node.			
VAR_INPUT			
	Execute	BOOL	On rising edge node information will be written.
	ConnectionHdl	DWORD	Connection handle.
	NodeHdl	DWORD	Node handle.
	NodeAddInfo	STRUCT	See 3.2.8 UANodeAdditionalInfo. Specifies the attribute and IndexRange.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
VAR_IN_OUT			
	Variable	Vendor specific	To be defined by vendor.

Notes: Vendors can handle “Variable” in a vendor specific way. Independent of the vendor specific solution the mapping of the controller data type and OPC-UA data type shall be handled in the function block. VAR_IN_OUT: „Variable” as would provide best type save solution for users: The client firmware is able to map the UA memory layout to the controller layout. The firmware client can receive the type definition from the UA-Server. Workaround would be to provide a byte array as “Variable” and the firmware client just provide the blob (UA memory layout – so called “raw data”) into that byte array. “Variable” could be the name of the variable so the internal firmware can get address, length, data type of variable.



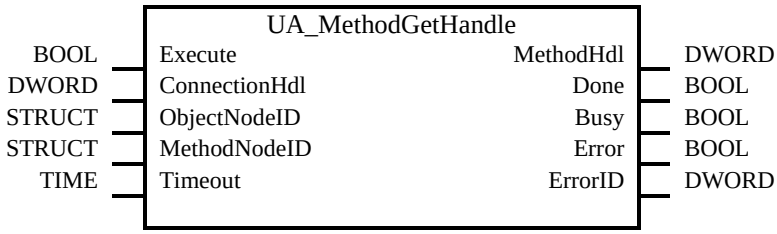
5.19. UA_WriteList

FB-Name	**UA_WriteList**	
This Function Block is used to write values to multiple nodes using a list of node handles.		
VAR_INPUT		
Execute	BOOL	On rising edge node values will be written.
ConnectionHdl	DWORD	Connection handle.
NodeHdlCount	UINT	Number of valid elements in the array to write.
NodeHdls	ARRAY OF DWORD	Array of Node Handles. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
NodeAddInfos	ARRAY OF STRUCT	See 3.2.8 UANodeAdditionalInfo. Array of UANodeAdditionalInfo. Specifies the attribute and IndexRange. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
Timeout	TIME	Time to response.
VAR_OUTPUT		
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB. Set to true if either ErrorID or any of the NodeErrorIDs indicates an error.
ErrorID	DWORD	Error code for the OPC UA service call.
NodeErrorIDs	ARRAY OF DWORD	Array of DWORD. Contains an error code for each valid element of the Variables array. Max length of array is to be defined by the vendor and shall be same size like the Variables array length.
VAR_IN_OUT		
Variables	ARRAY OF Vendor specific	Vendor specific. Max length of array is to be defined by the vendor.
Notes: Vendors can handle “Variables” in a vendor specific way. Independent of the vendor specific solution the mapping of the controller data type and OPC-UA data type shall be handled in the function block. VAR_IN_OUT: „Variables” as would provide best type save solution for users: The client firmware is able to map the UA memory layout to the controller layout. The firmware client can receive the type definition from the UA-Server. Workaround would be to provide a byte array as “Variables” and the firmware client just provide the blob (UA memory layout – so called “raw data”) into that byte array. “Variables” could be the name of the variable so the internal firmware can get address, length, data type of variable.		



5.20. UA_MethodGetHandle

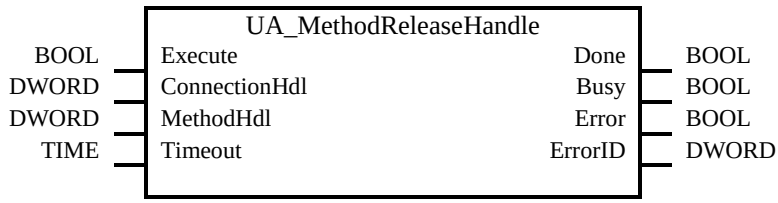
FB-Name	**UA_MethodGetHandle**	
This Function Block is used to get the method handle for a method call.		
VAR_INPUT		
Execute	BOOL	FB performs its task on rising edge on this input.
ConnectionHdl	DWORD	Connection handle.
ObjectNodeID	STRUCT	See 3.2.3 UANodeID.
MethodNodeID	STRUCT	See 3.2.3 UANodeID.
Timeout	TIME	Time to response.
VAR_OUTPUT		
MethodHdl	DWORD	Method handle.
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
Notes:		



5.21. UA_MethodReleaseHandle

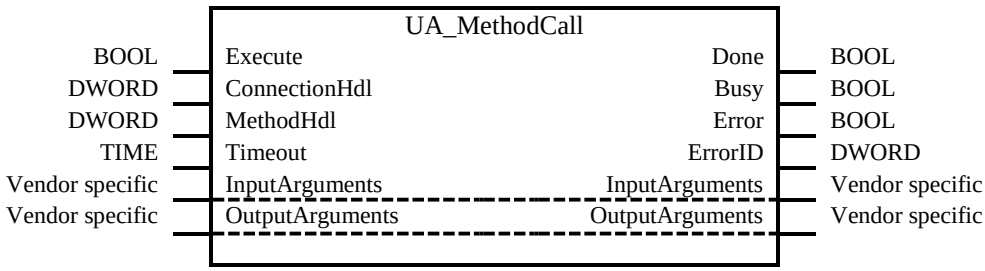
FB-Name	**UA_MethodReleaseHandle**	
This Function Block is used to release the method handle.		
VAR_INPUT		
Execute	BOOL	FB performs its task on rising edge on this input.
ConnectionHdl	DWORD	Connection handle.
MethodHdl	DWORD	Method handle to be released.
Timeout	TIME	Time to response.
VAR_OUTPUT		
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.

Notes: After calling UA_MethodReleaseHandle the MethodHdl will be invalid.



5.22. UA_MethodCall

FB-Name	**UA_MethodCall**	
This Function Block is used to call a method routine.		
VAR_INPUT		
Execute	BOOL	FB performs its task on rising edge on this input.
ConnectionHdl	DWORD	Connection handle.
MethodHdl	DWORD	Method handle.
Timeout	TIME	Time to response.
VAR_OUTPUT		
Done	BOOL	FB has completed its task.
Busy	BOOL	The FB is not finished and new output values are to be expected.
Error	BOOL	Signals that an error has occurred within the FB.
ErrorID	DWORD	Error code.
VAR_IN_OUT		
InputArguments	Vendor specific	Variable containing input parameters. Vendor specific.
OutputArguments	Vendor specific	Variable containing output parameters. Vendor specific.
Notes: -		



6. Diagnosis

6.1. UA_ConnectionGetStatus

FB-Name		**UA_ConnectionGetStatus**	
This Function Block is used to get the connection status.			
VAR_INPUT			
	Execute	BOOL	FB performs its task on rising edge on this input.
	ConnectionHdl	DWORD	Connection handle.
	Timeout	TIME	Time to response.
VAR_OUTPUT			
	ConnectionStatus	ENUM	See 3.1.9 UAConnectionStatus. The outputs ServerState and ServiceLevel are only valid if the ConnectionStatus is UAConnectionStatus_Connected.
	ServerState	ENUM	See 3.1.10 UAServerState. The ServerState is UAServerState_UNKOWN if the ConnectionStatus is not UAConnectionStatus_Connected.
	ServiceLevel	BYTE	ServiceLevel describes the ability of the Server to provide its data to the client. The value range is from 0 to 255, where 0 indicates the worst and 255 indicates the best. The intent is to provide the clients an indication of availability among redundant Servers.
	Done	BOOL	FB has completed its task.
	Busy	BOOL	The FB is not finished and new output values are to be expected.
	Error	BOOL	Signals that an error has occurred within the FB.
	ErrorID	DWORD	Error code.
Notes: -