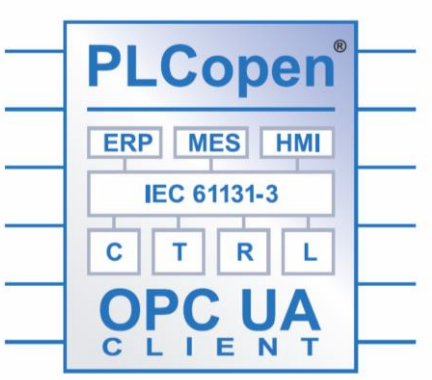
PLCopen®
for efficiency in automation

PLCopen OPC UA Client for IEC 61131-3

Scope



This specification was created by a joint working group of the OPC Foundation and PLCopen. It defines a set of IEC 61131-3 based function blocks for mapping the OPC UA Client functionalities. With this functionality implemented on a controller it becomes possible to initiate a communication session to any other available OPC UA Server. This is flipping the classical CIM Triangle where, 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.

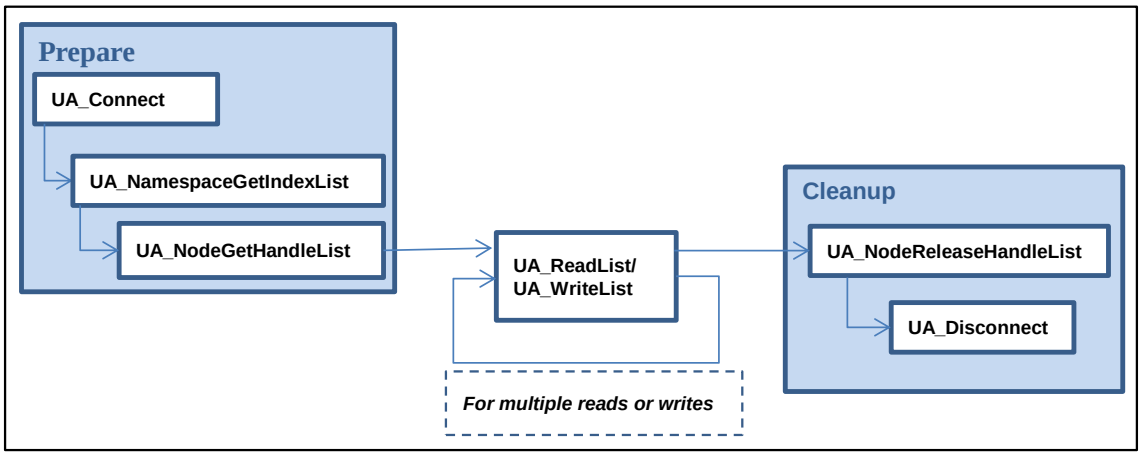
In a modern smart network, every device or service is able to initiate independent communication with all other services.
The OPC-UA client functionality out of the IEC61131-3 controller makes it possible that a controller can exchange complex data structures horizontally with other controllers independently 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.

Notes:
1. 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.
2. The function blocks are based on the second Edition of IEC61131-3.
3. Within the PLCopen OPC UA Client specification a lot of derived, enumerated and structured datatypes are defined, as well as user specific datatypes identified, all of which are used by the function blocks but nor further described here. Also the defined error codes are not described here.

The basic sequences for communication

A whole set of function blocks have been defined. But in order to perform an operation like read a list, write a list or do invoke a method call, one has to prepare the communication following the sequence of calls as described hereunder. After finalization, one has to stop the communication and clean up.

Read and Write of multiple items
UA_Connect is used to create an (optional secure) transport connection of an OPC-UA session. UA_Connect is to be performed once for each connection. The UA_NamespaceGetIndexList 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 communication is done, the node handle is not required anymore and shall be released via the use of UA_NodeReleaseHandleList for all relevant handles. The connection handle shall be released using UA_Disconnect.
A list is handled as an array of the related base type (e.g. UANodeID or UANodeAdditionalInfo). Additionally, there is a length specified which holds the number of elements in the array.
The UA_NodeGetHandleList will return a UANodeHdl array. This call will not verify that the given UANodeID is valid. It will just be checked if it is structurally right, otherwise an error in the corresponding error element (NodeErrorIDs) will be returned. The output array of UA_NodeGetHandleList can be used unchanged for subsequent calls to function blocks UA_ReadList, UA_WriteList, but the control implementation shall check always the corresponding error element (NodeErrorIDs). In case of any general error no outputs shall be changed from the underlying implementation.



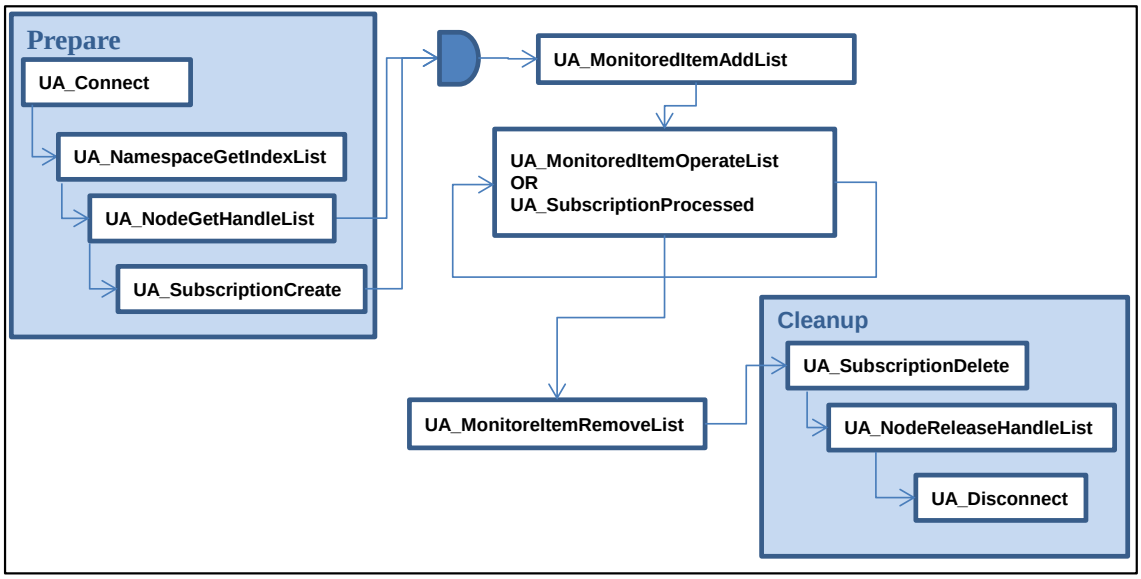
Subscription and Monitored Items

Monitoring of nodes does invert the communication interaction: the control program is initiating the communication but as a consequence the values will be pushed from the UA-Server to the control program.
There are two modes to actually retrieve latest values within the control program:

- **Controller-sync:** The control program can decide when values are updated. The control program polls internally if the UA-Client-Firmware received updates;
- **FW-sync:**
  The firmware could internally update the values of the memory of the controller and magically the value are just updated from outside.

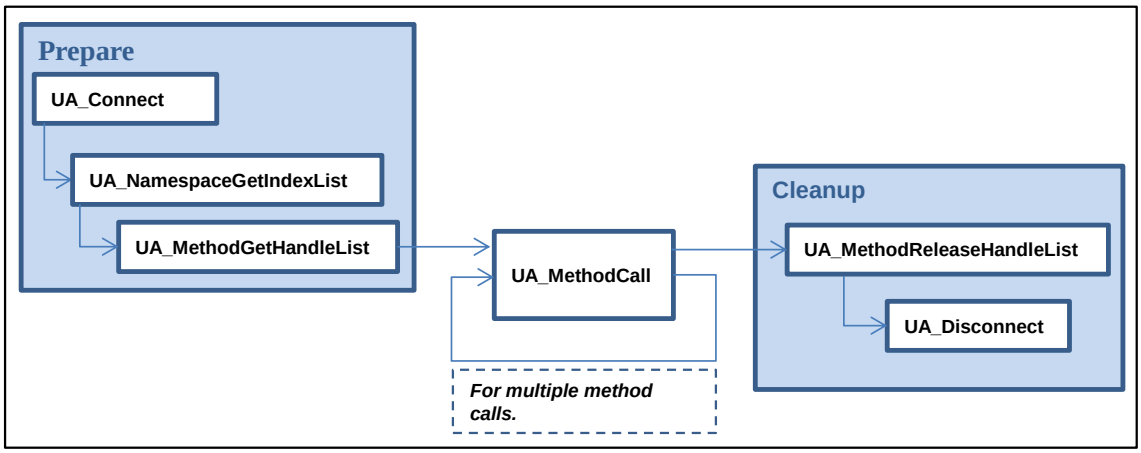
The following function blocks are used to create subscriptions and to add monitored items to this subscription. To create a subscription, a valid connection handle is required. The connection handle is to be acquired using UA_Connect once.

UA_SubscriptionCreate will create a subscription, and needs to be called for every subscription needed. The applicable SubscriptionHdl will be returned on successful execution of the function block UA_SubscriptionCreate. In order to monitor an item, a NodeHdl for that specific node is required. In other words, both the applicable UA_SubscriptionCreate and UA_NodeGetHandleList are to be called before calling the related UA_MonitoredItemAddList. UA_MonitoredItemAddList is used to add items to a subscription identified by a SubscriptionHdl. The items to be monitored are to be assigned to this FB in form of NodeHdl. UA_MonitoredItemModifyList can be used to modify monitoring settings like sampling interval, deadband type, and deadband value and hence it can be called optionally.
Take note to delete the subscription. Release the NodeHdl before you disconnect. Unless UA_SubscriptionDelete is called the Subscription will continue working, even if UA_NodeReleaseHandleList is called.



Using Method Calls

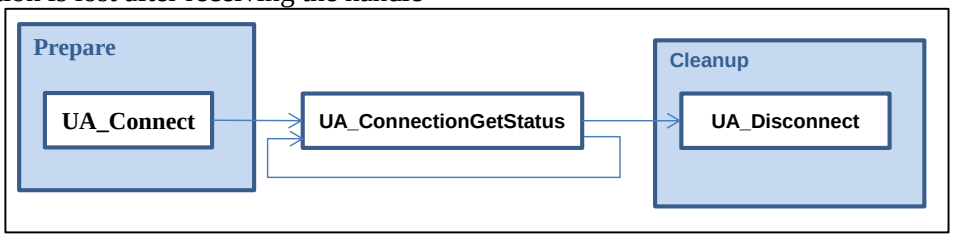
Method calls are a way to perform a remote call on the called object, which can run on a different machine or be linked to any server, which than replies with the result to the client. The appropriate sequence for initiating a method call is shown below. A valid method handle is necessary to call a method. Successful call of UA_Method
GetHandleList will deliver a valid MethodHdls. One shall release the method handle list before you disconnect.



Diagnostics

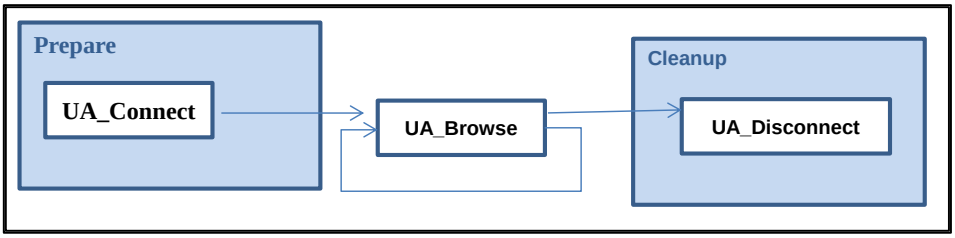
This procedure is to check if the connection is still alive. The function block UA_Connect will deliver the ConnectionHdl. UA_ConnectionGetStatus requires this ConnectionHdl as input to deliver the connection status. In case the connection is lost after receiving the handle and while calling the UA_ConnectionGetStatus, ServerState Unknown will be returned.

NOTE: It is recommended to call UA_ConnectionGetStatus periodically but for performance reasons not in every control program cycle.



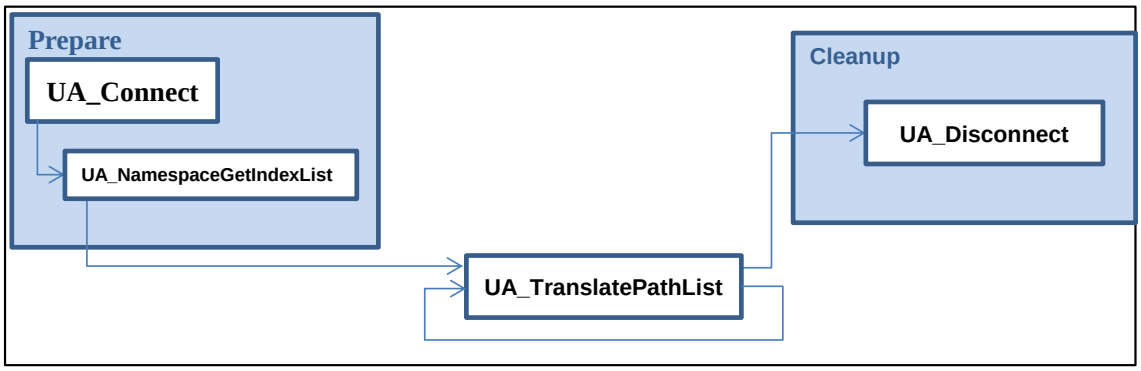
Browsing

Browsing is used by a client to navigate through the Address Space of an OPC-UA Server. By passing a starting node the server returns a list of nodes by references.
To be able to browse a valid connection handle is required. Function block UA_Connect will deliver the ConnectionHdl. UA_Browse takes a structure for starting Node description and filter criteria. The result is an array of structures for references and target Nodes.
With browsing one can "ask", e.g. for validation purposes, for the structure and all elements that are running in a server (as far as the client has the appropriate access rights).



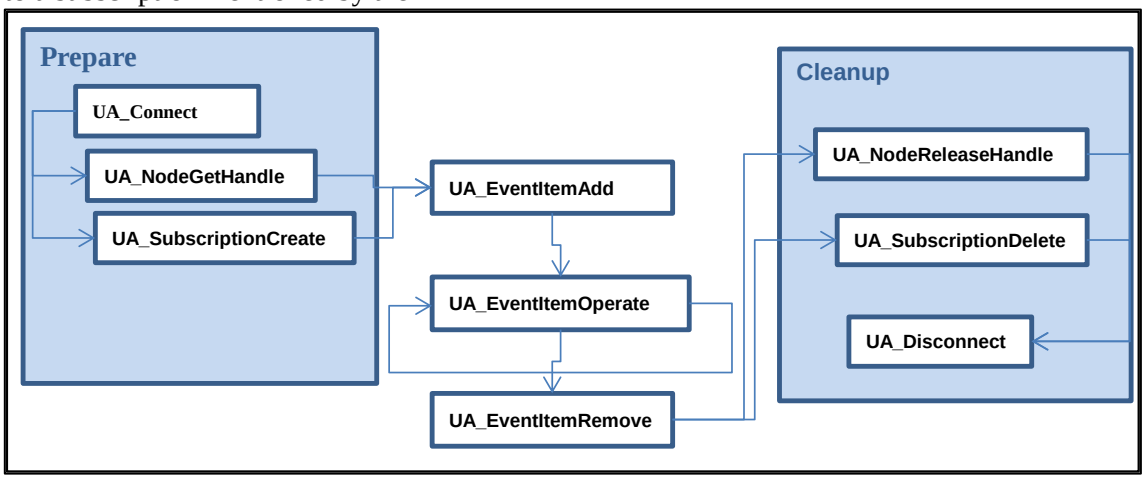
TranslatePath

This function block is used to request that the Server translates one or more UABrowsePaths to UANodeIDs. Each UA_BrowsePath is constructed of a starting UANodeID and a UARelativePath. The specified starting UANodeID identifies the UANodeID from which the UARelativePath is based. The UARelativePath contains a sequence of UARelativePathElement and UAQualifiedName.
One purpose of this function block is to allow programming against type definitions. For example, an ObjectType "Boiler" may have a "HeatSensor" Variable as InstanceDeclaration. A graphical element programmed against the "Boiler" may need to display the Value of the "HeatSensor". If the graphical element would be called on "Boiler1", an instance of "Boiler", it would need to call this Service specifying the UANodeID of "Boiler1" as starting UANodeID and the UABrowsePaths of the "HeatSensor" as browse path. The function block would return the UANodeID of the "HeatSensor" of "Boiler1" and the graphical element could subscribe to its value.
If an OPC UA Node has multiple targets with the same UABrowsePaths, the underlying server will return a list of UANodeIDs. However, since one of the main purposes of this function block is to support programming against type definitions, the UANodeID of the OPC UA Node based on the type definition of the starting OPC UA Node is returned as the first UANodeID in the list.



Monitor Events

A typical OPC-UA Server can be configured to fire Events to a Client. OPC-UA specifies a wide range of different Events. OPC-UA Clients can receive Events when subscribing to an Event Notifier.
In order to monitor an item, a NodeHdl for that specific node is required. Both UA_SubscriptionCreate and UA_NodeGetHandle are to be called before calling UA_EventItemAdd. UA_EventItemAdd is used to add an event item to a subscription mentioned by the SubscriptionHdl. The node of which events are monitored is to be assigned to this FB in form of NodeHdl. UA_EventItemOperate can be used to get information about the incoming events occurred. 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.



Overview of the defined function blocks:

UA_CONNECT	UA_DISCONNECT
UA_NAMESPACEGETINDEXLIST	UA_TRANSLATEPATHLIST
UA_SERVERGETURIBYINDEX	UA_SERVERGETINDEXBYURILIST
UA_NODEGETHANDLELIST	UA_NODERELEASEHANDLELIST
UA_NODEGETINFORMATION	
UA_SUBSCRIPTIONCREATE	UA_SUBSCRIPTIONDELETE
UA_SUBSCRIPTIONMODIFY	UA_SUBSCRIPTIONPROCESSED
UA_MONITOREDITEMADDLIST	UA_MONITOREDITEMREMOVELIST
UA_MONITOREDITEMMODIFYLIST	UA_MONITOREDITEMOPERATELIST
UA_READLIST	UA_WRITELIST
UA_METHODGETHANDLELIST	UA_METHODRELEASEHANDLELIST
UA_METHODCALL	UA_BROWSE
UA_EVENTITEMADD	UA_EVENTITEMOPERATELIST
UA_EVENTITEMREMOVELIST	UA_HISTORYUPDATE
For diagnosis	UA_CONNECTIONGETSTATUS

The specification also includes the basis for a compliance statement.
For the full technical file check www.PLCopen.org