SIGMATEK

LASAL OPC-UA-Connector

Date of creation: 26.08.2016 | Version date: 21.10.2016 | Article number: XX-XXX-XXX-E

Publisher: SIGMATEK GmbH & Co KG
A-5112 Lamprechtshausen
Tel.: 06274/4321
Fax: 06274/4321-18
Email: office@sigmatek.at
WWW.SIGMATEK-AUTOMATION.COM

Translation from German

Contents

1 OPC-UA Introduction

1.1 What is OPC-UA?

OPC Unified Architecture, short OPC-UA, is an industrial communication protocol.

As the newest of all OPC specifications of the OPC Foundation, OPC-UA differs considerably from its predecessors, mainly in its capability to not only transfer but also machine-readably semantically describe machine data (control variables, measurement values, parameters, etc.).

The OPC-UA participants can be controls, master computers, ERP systems, and many others.

With the SIGMATEK OPC-UA class the following data can be transferred without much programming effort:

- Transfer of simple data types: DINT, UDINT, REAL and STRING
- OPC-UA client data transfer to external OPC-UA servers
- File transfer from client to server and vice versa

1.2 Contents of Delivery

OPC-UA class and sticker

Article number OPC UA Embedded License (in the form of license sticker) 02-010-074

1.3 Placement

The license sticker has to be applied next to the type label of the hardware, where the OPC-UA software is installed.

1.4 Supported OPC UA Services

- FindServer, GetEndpoints
- CreateSession, ActivateSession, CloseSession
- Browse, Translate
- Read
- Write
- CALL
- CreateSubscription, ModifySubscription, DeleteSubscription
- CreateMonitoredItems, ModifyMonitoredItems, DeleteMonitoredItems
- Publish, Republish

1.5 Supported OPC UA Features and Profiles

1.5.1 General

- Standard UA Server

1.5.2 Data Access

- DataAccess Server Facet
- ComplexType Server Facet

1.5.3 Events

- Basic Event Subscription Server Facet
- Address Space Notifier Server Facet

1.5.4 Methods

- Method Server Facet

1.5.5 Alarms & Conditions

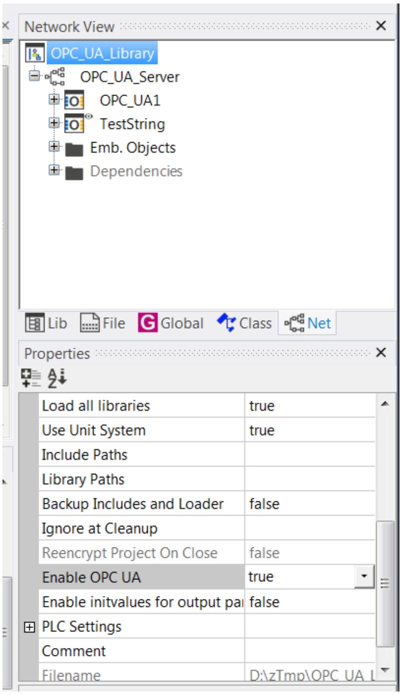
- A&C Simple Server Facet
- A&C Address Space Instance Server Facet
- A&C Enable Server Facet
- A&C Alarm Server Facet
- A&C Acknowledgeable Alarm Server Facet
- A&C Exclusive Alarming Server Facet
- A&C Non-Exclusive Alarming Server Facet

2 Lasal Project … Necessary First Steps

A new Lasal project in the beginning is not able to establish OPC-UA communication.

- In each station that should work as an OPC-UA server first the described OPC-UA class has to be imported and placed in a network.

- Additionally OPC-UA has to be enabled in the project!



Load all libraries	true
Use Unit System	true
Include Paths	
Library Paths	
Backup Includes and Loader	false
Ignore at Cleanup	
Reencrypt Project On Close	false
Enable OPC UA	true
Enable initvalues for output pa	false
PLC Settings	
Comment	
Filename	D:\zTmp\OPC_UA_L

3 Simple Data Exchange

This chapter deals with the exchange of base data types.

The normal data exchange in SIGMATEK systems is realized with server variables of one or more classes.

Supported at the moment: DINT, UDINT, REAL and STRING

Accessing the SIGMATEK OPC-UA server is done via its endpoint Url.

For communication port 4842 is used.

Example for endpoint Url: `opc.tcp://10.10.160.41:4842`

3.1 Principle

- The user has to declare all OPC-UA variables in the PLC project as such. It makes sense to create an own class for the OPC-UA variables and place it in a network. See explanation for implementing below.
- All variables to be transferred are entered in a XML file. This is done fully automatically by the SIGMATEK system! The XML file is transferred to the CPU while downloading the project.
- When booting the PLC the OPC-UA server reads and interprets the XML file and from this generates an OPC-UA address space.
- The OPC-UA server is available for the outside world, so an OPC-UA client can connect to the server and read and write all OPC-UA variables and their properties.

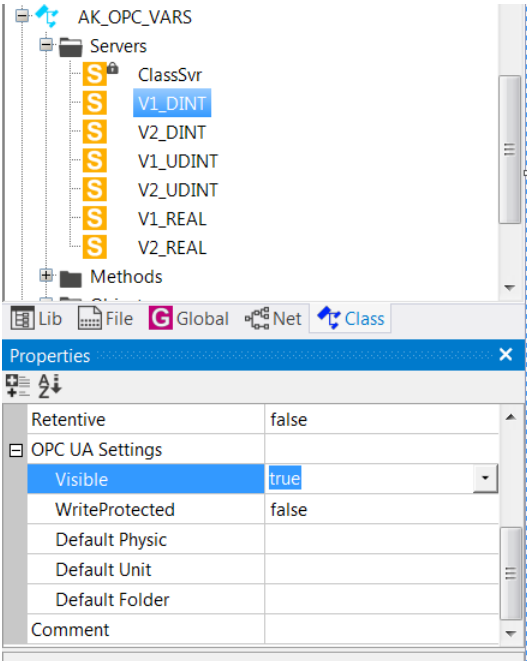
3.2 Implementing in the PLC Project

As mentioned before, the normal data exchange in SIGMATEK systems is realized with server variables (types DINT, UDINT, REAL and STRING).

The SIGMATEK system generates the OPC_UA.XML file for the OPC server.

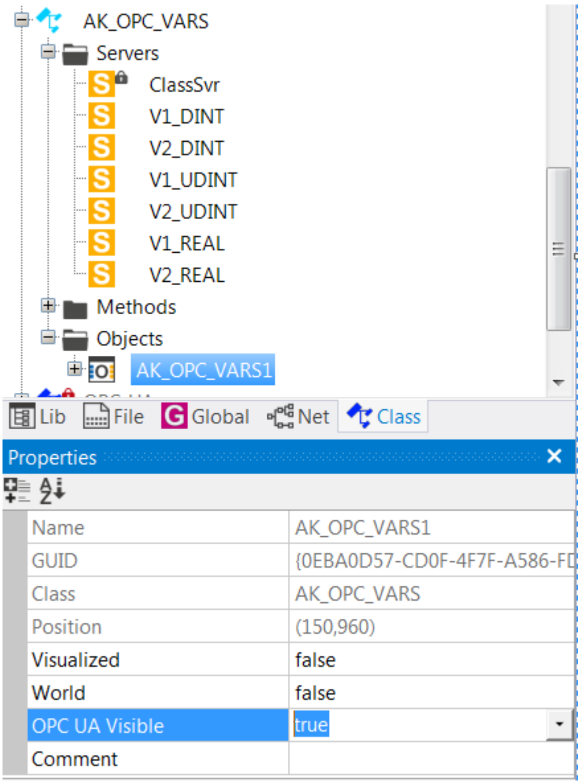
Here the following requirements have to be met:

- in the properties of all desired **servers** the OPC attribute “Visible” has to be “true”
- the OPC attribute “WriteProtected” depends on the application



Retentive	false
OPC UA Settings	
Visible	true
WriteProtected	false
Default Physic	
Default Unit	
Default Folder	
Comment	

- in all instanced **objects** containing servers of OPC-UA data transfer, the attribute “**OPC UA Visible**” has to be “**true**”



Name	AK_OPC_VARS1
GUID	{0EBA0D57-CD0F-4F7F-A586-FD
Class	AK_OPC_VARS
Position	(150,960)
Visualized	false
World	false
OPC UA Visible	true
Comment	

- transfer project to the PLC

The XML file is generated when compiling the program and transferred to the control only with the download. When resetting/starting the CPU the file is neither created nor changed.

Now the OPC-UA client should be able to access the variables.

The program UaExpert can be used to test the program (see according chapter in this documentation).

4 Configuration Using an XML File

One (or more) user-specific configuration files are needed to inform the OPC UA server of the corresponding data points (variables), including attributes and directories.

As mentioned before, Lasal Class 2 automatically generates the OPC_UA.XML file.

When downloading the project, this XML file is transferred to the CPU.

However, a manually generated XML file can help to enable extended functionality.

This XML file must be stored in the corresponding PLC, in which the OPC-UA server is running. An OPC-UA client can only access data points configured this way!

As a default this file is named “OPC_UA.xml”, it is placed in the root directory of the control. With overwriting the virtual method OPC_UA::FunctSetUp name and path can be changed.

4.1 General Structure

OPC_UA.XML

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
    <Trace TraceLevel="48" />
    <Release>
        <ReleasePath Path="C:\OPCUA\"/>
        <ReleasePath Path="E:\OPCUA\"/>
    </Release>
    <DataSet>
        <DataElement Hostname="ClassSvr"       Type="DINT"    Writeprotected="true"  Physic="" Unit="" Folder="M01\Analyse"   Label="MyTask1.ClassSvr"/>
        <DataElement Hostname="ErrorCode"      Type="DINT"    Writeprotected="true"  Physic="" Unit="" Folder="M01\Analyse"   Label="MyTask1.ErrorCode"/>
        <DataElement Hostname="CycleCounter"   Type="DINT"    Writeprotected="true"  Physic="" Unit="" Folder="M01\Analyse"   Label="MachineData.CycleCounter"/>
        <DataElement Hostname="Test32"         Type="DINT"    Writeprotected="false" Physic="" Unit="" Folder="M01\Analyse"   Label="MachineData.Test32"/>
        <DataElement Hostname="TestString"     Type="STRING"  Writeprotected="false" Physic="" Unit="" Folder="M01\Analyse"   Label="TestString.Data"/>
    </DataSet>
</Config>
```

All configurations are mad in the "Config" tag. Here the sub tags "Trace", "Release", "Server" and "DataSet” are distinguished.

Trace	Element is optional. It defines the level for trace outputs. For possible values see the class description OPC-UA, area server - TraceLevel. Tag "Trace" attribute TraceLevel defines the trace level for trace outputs
Release	defines one or more shared directories for file up-/download Tag "ReleasePath" attribute Path defines the possible path(s) for file transfers
Servers	Element is optional. It defines one or more “external” OPC-UA servers, that can be accessed by the “own” OPC-UA class as clients. See chapter “Client Data Transfer … "
DataSet	defines the individual data points in the PLC with the corresponding attributes. OPC-UA clients can access these data points! Tag "DataElement" with attributes Type (DINT, UDINT, REAL, STRING) Hostname data element name to write "Writeprotected" (true, false) Physic user specific text - optional Unit user specific text - optional Folder user specific folder - optional Label actual name of the data element (Object.Server) There also are additional attributes for the definition of the nodes in the “external” OPC-UA servers in case of the optional client data transfer. See chapter “Client Data Transfer … "

4.2 Example for OPC-UA Variables with 2 Configuration Files

Config1.xml

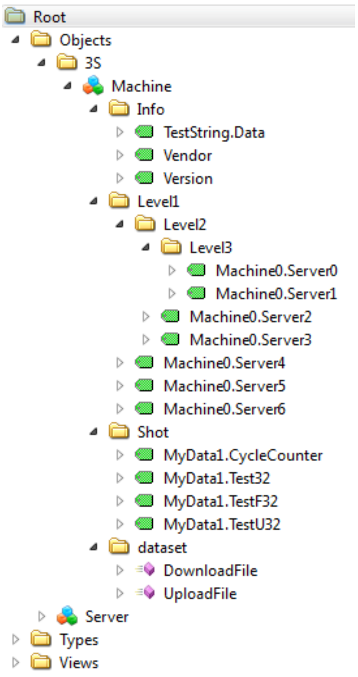
```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config Version="1.0">
    <DataSet>
        <DataElement Hostname="Vendor"     Type="STRING"   Writeprotected="true"  Physic=""   Unit=""     Folder="Info"       Label="Vendor"           />
        <DataElement Hostname="Version"    Type="STRING"   Writeprotected="true"  Physic=""   Unit=""     Folder="Info"       Label="Version"          />
        <DataElement Hostname="TestString" Type="STRING"   Writeprotected="false"  Physic=""  Unit=""     Folder="Info"       Label="TestString.Data" />
    </DataSet>
</Config>
```

Config2.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config Version="1.0">
    <Release>
        <ReleasePath Path="C:\OPCUA\"/>
        <ReleasePath Path="E:\OPCUA\"/>
    </Release>
    <DataSet>
        <!-- Comment -->
        <DataElement Hostname="Counter" Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Shot"                    Label="MyData1.CycleCounter"    />
        <DataElement Hostname="Test32"  Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Shot"                    Label="MyData1.Test32"          />
        <DataElement Hostname="TestU32" Type="UDINT"    Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Shot"                    Label="MyData1.TestU32"         />
        <DataElement Hostname="TestF32" Type="REAL"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Shot"                    Label="MyData1.TestF32"         />

        <DataElement Hostname="M0_00"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1\Level2\Level3"    Label="Machine0.Server0"        />
        <DataElement Hostname="M0_01"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1\Level2\Level3"    Label="Machine0.Server1"        />
        <DataElement Hostname="M0_02"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1\Level2"           Label="Machine0.Server2"        />
        <DataElement Hostname="M0_03"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1\Level2"           Label="Machine0.Server3"        />
        <DataElement Hostname="M0_04"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1"                  Label="Machine0.Server4"        />
        <DataElement Hostname="M0_05"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1"                  Label="Machine0.Server5"        />
        <DataElement Hostname="M0_06"   Type="DINT"     Writeprotected="false"  Physic=""   Unit="SEC"  Folder="Level1"                  Label="Machine0.Server6"        />
    </DataSet>
</Config>
```

Address Space of the Server

The two configuration files shown above define the following address space:



```
Root
  Objects
    3S
      Machine
        Info
          TestString.Data
          Vendor
          Version
        Level1
          Level2
            Level3
              Machine0.Server0
              Machine0.Server1
            Machine0.Server2
            Machine0.Server3
          Machine0.Server4
          Machine0.Server5
          Machine0.Server6
        Shot
          MyData1.CycleCounter
          MyData1.Test32
          MyData1.TestF32
          MyData1.TestU32
        dataset
          DownloadFile
          UploadFile
    Server
  Types
  Views
```

5 Client Data Transfer with External OPC-UA Servers

This client function should not be confused with a full OPC-UA client.

It is possible to synchronize variable values of one or more external OPC-UA servers with the data of the local OPC-UA server.

This client functionality is integrated in the OPC-UA class. Thus it is possible to access data points of “external” OPC-UA clients reading or writing.

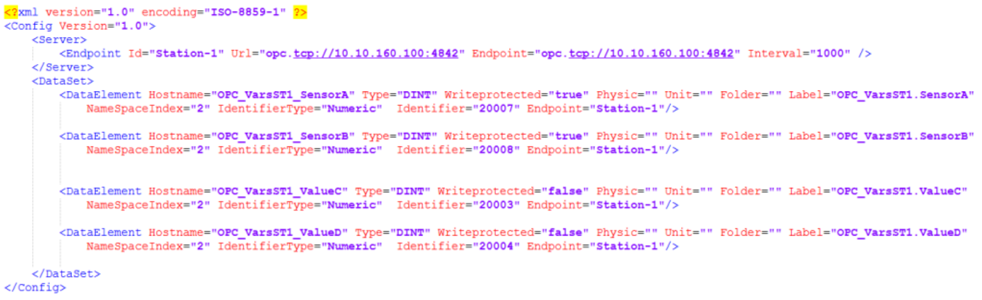
The configuration is also realized with a XML file.

Theoretically this configuration can be entered in the standard file (OPC_UA.xml) - but it is not recommended because of a better overview.

During the boot phase the OPC-UA class also searches for the XML file for the client data exchange.

The name is OPC_UA_Client.xml.

Example:



```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
    <Server>
        <Endpoint Id="Station-1" Url="opc.tcp://10.10.160.100:4842" Endpoint="opc.tcp://10.10.160.100:4842" Interval="1000" />
    </Server>
    <DataSet>
        <DataElement Hostname="OPC_VarsST1_SensorA" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorA"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20007" Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_SensorB" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorB"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20008" Endpoint="Station-1"/>


        <DataElement Hostname="OPC_VarsST1_ValueC" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueC"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20003" Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_ValueD" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueD"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20004" Endpoint="Station-1"/>

    </DataSet>
</Config>
```

All configurations are mad in the "Config" tag. Hereby the sub tags "Server" and "DataSet” are needed.

Servers	This element defines one or more “external” OPC-UA servers to be accessed by the OPC-UA class as a client. The tag “Endpoint” defines the connection properties to the external server. Id — unique identifier for the OPC-UA server Url — Url of the remote OPC-UA server Endpoint — end point of the connection to be used Interval — interval in [ms] for the cyclic data exchange
DataSet	defines the individual data points in the PLC with the corresponding attributes. Tag "DataElement" with attributes The items in the **first line** are the same as those of the standard OPC-UA data points - they define the variables of the **own station**: Type — (DINT, UDINT, REAL, STRING) Hostname — data element name to write Writeprotected — true = READING from an external station false = WRITING to an external station Physic — [user specific text] - optional Unit — [user specific text] - optional Folder — [user specific folder] - optional Label — actual name of the data element (Object.Server) The additional attributes in the **second line** serve for the definition of the nodes in the **external OPC-UA servers** for the case of the optional client function: NameSpaceIndex — name space where the identifier of the node is located IdentifierType — type of the identifier ("Numeric" or "String") Identifier — unique ID (if "IdentifierType" = "Numeric") Name of the identifier (if "IdentifierType" = "String") Endpoint — ID of the server connection to be used (see sub tag "Server")

Note

The data synchronization of a variable can only work in one direction.

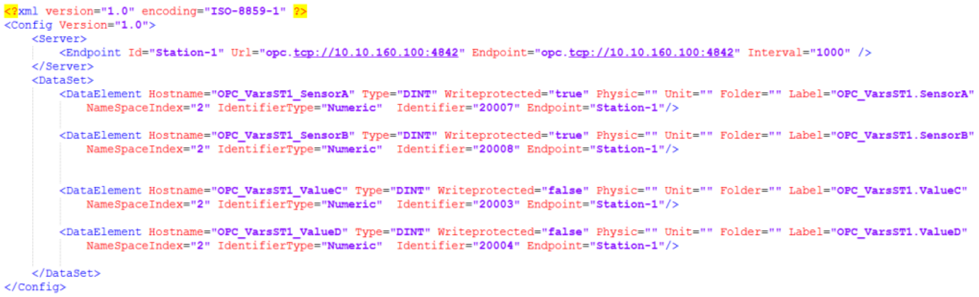
This is defined in the attribute “Writeprotected”.

Writeprotected="true" …	data are read from the external server
Writeprotected="false" …	data are written to the external server

Access to the data of the external server can be realized in two ways:

1: IdentifierType "Numeric"

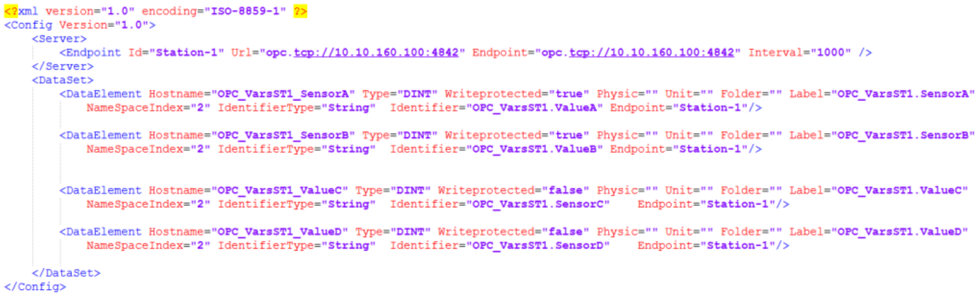
= identify variable on the external server with unique identifier number



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
    <Server>
        <Endpoint Id="Station-1" Url="opc.tcp://10.10.160.100:4842" Endpoint="opc.tcp://10.10.160.100:4842" Interval="1000" />
    </Server>
    <DataSet>
        <DataElement Hostname="OPC_VarsST1_SensorA" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorA"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20007" Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_SensorB" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorB"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20008" Endpoint="Station-1"/>


        <DataElement Hostname="OPC_VarsST1_ValueC" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueC"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20003" Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_ValueD" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueD"
            NameSpaceIndex="2" IdentifierType="Numeric"  Identifier="20004" Endpoint="Station-1"/>

    </DataSet>
</Config>
```

2: IdentifierType "String"

= identify variable on the external server with unique identifier name



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
    <Server>
        <Endpoint Id="Station-1" Url="opc.tcp://10.10.160.100:4842" Endpoint="opc.tcp://10.10.160.100:4842" Interval="1000" />
    </Server>
    <DataSet>
        <DataElement Hostname="OPC_VarsST1_SensorA" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorA"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.ValueA" Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_SensorB" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorB"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.ValueB" Endpoint="Station-1"/>


        <DataElement Hostname="OPC_VarsST1_ValueC" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueC"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.SensorC"    Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_ValueD" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueD"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.SensorD"    Endpoint="Station-1"/>

    </DataSet>
</Config>
```

The property "Numeric" or "String" have to be set accordingly on the **REMOTE STATION**!

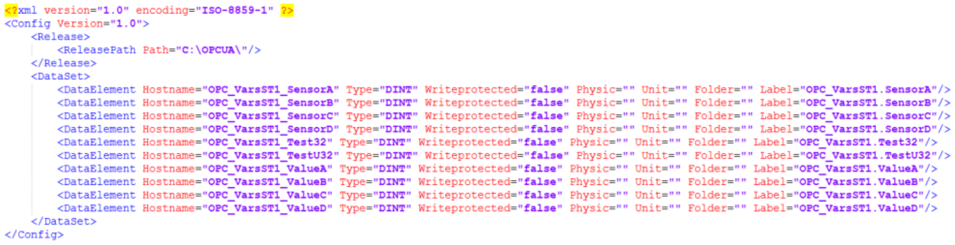
In case of the SIGMATEK OPC-UA class this is done with the help of the client connection "Config".

For overview reasons, it is recommended to set all participants in the same way.

Example

In our example the same program with an OPC-UA server is running on two controls.

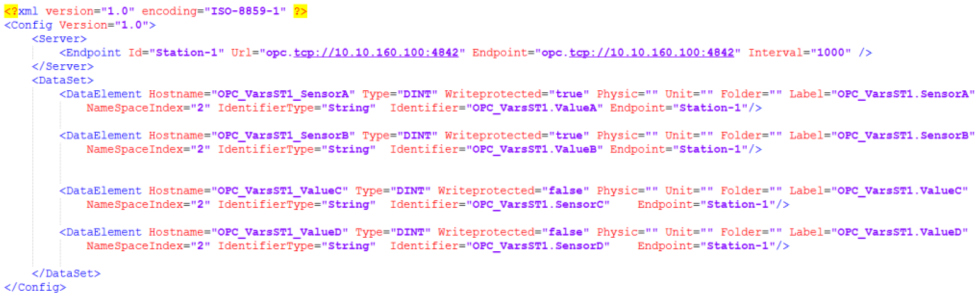
Because of the configuration in Lasal Class 2 this results in the following XML file:



```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
    <Release>
        <ReleasePath Path="C:\OPCUA\"/>
    </Release>
    <DataSet>
        <DataElement Hostname="OPC_VarsST1_SensorA" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorA"/>
        <DataElement Hostname="OPC_VarsST1_SensorB" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorB"/>
        <DataElement Hostname="OPC_VarsST1_SensorC" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorC"/>
        <DataElement Hostname="OPC_VarsST1_SensorD" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorD"/>
        <DataElement Hostname="OPC_VarsST1_Test32" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.Test32"/>
        <DataElement Hostname="OPC_VarsST1_TestU32" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.TestU32"/>
        <DataElement Hostname="OPC_VarsST1_ValueA" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueA"/>
        <DataElement Hostname="OPC_VarsST1_ValueB" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueB"/>
        <DataElement Hostname="OPC_VarsST1_ValueC" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueC"/>
        <DataElement Hostname="OPC_VarsST1_ValueD" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueD"/>
    </DataSet>
</Config>
```

One of the controls should also work as a client.

There additionally the file OPC_UA_Client.XML has to be created/stored manually.



```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
    <Server>
        <Endpoint Id="Station-1" Url="opc.tcp://10.10.160.100:4842" Endpoint="opc.tcp://10.10.160.100:4842" Interval="1000" />
    </Server>
    <DataSet>
        <DataElement Hostname="OPC_VarsST1_SensorA" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorA"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.ValueA" Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_SensorB" Type="DINT" Writeprotected="true" Physic="" Unit="" Folder="" Label="OPC_VarsST1.SensorB"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.ValueB" Endpoint="Station-1"/>


        <DataElement Hostname="OPC_VarsST1_ValueC" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueC"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.SensorC"    Endpoint="Station-1"/>

        <DataElement Hostname="OPC_VarsST1_ValueD" Type="DINT" Writeprotected="false" Physic="" Unit="" Folder="" Label="OPC_VarsST1.ValueD"
            NameSpaceIndex="2" IdentifierType="String"  Identifier="OPC_VarsST1.SensorD"    Endpoint="Station-1"/>

    </DataSet>
</Config>
```

The remote station (OPC-UA server) was named “Station-1”.

After a reset/run of the client CPU the following data transfer runs:

… reading the OPC Var. ValueA of "Station-1" & saving in own OPC Var. SensorA

… reading the OPC Var. ValueB of "Station-1" & saving in own OPC Var. SensorB

… reading the own OPC Var. ValueC & saving in OPC Var. SensorC on "Station-1"

… reading the own OPC Var. ValueD & saving in OPC Var. SensorD on "Station-1"

6 OPC-UA Functions

6.1 Reading/Writing Variables

In this documentation reading and writing of variables has already be dealt with.

All servers of a LASAL project can be provided via configuration to the OPC UA server. Through the configuration, which variables can be read or written by the user is set.

Currently, the data types DINT, UDINT, REAL and STRING.

Recommendation

Since access to the data entries from the OPC UA server cannot be triggered by user specifications, an appropriate interface between the OPC UA server and PLC is recommended.

6.2 MonitoredItems

For each OPC-UA variable that can be read by the client, a MonitoredItem can also be created on the client side.

The monitoring of such items however works on the server side.

Recommendation

Each monitored item must be internally monitored by the server for changes, which increases the load on the entire system.

There should only be as many MonitoredItems created as absolute necessary.

As rule, there is one trigger variable in the entire system. Using this variable, whether or not data has changed can be detected (e.g. cycle counter).

A MonitoredItem should be created for this field, so that the client can be automatically informed of changes by the server with an event. After a change trigger change, the client read the changed data with a single read process.

6.3 File Download

With this standard OPC-UA method, a single file can be sent from the OPC-UA client to the OPC-UA server.

No background knowledge concerning the server application is needed.

CLIENT: StatusCode = DownloadFile([input]String Filename, [input]ByteString Data)

Filename	Name, under which the file is stored for on the PLC. Optionally an absolute path can be given with the file name. If no path is defined, the path "C:\OPCUA\" is used. When entering an absolute path, it is validated according to the shared paths in the configuration. Only paths from the configuration and "C:\OPCUA" are valid.
Data	Contains the file content in the form of a byte string.
StatusCode	Return value indicates the success / failure of the method call.

6.4 File Upload

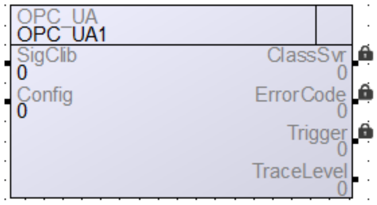
With this standard OPC-UA method, a single files can be read from the OPC-UA server and sent to the OPC-UA client.

No background knowledge concerning the server application is needed.

CLIENT: StatusCode = UploadFile([input]String Filename, [output]ByteString Data)

Filename	name, under which the file is searched for on the PLC. Optionally an absolute path can be given with the file name. If no path is defined, the path "C:\OPCUA\" is used. When entering an absolute path, it is validated according to the shared paths in the configuration. Only paths from the configuration and "C:\OPCUA" are valid.
Data	Contains the file content in the form of a byte string.
StatusCode	Return value indicates the success / failure of the method call.

7 OPC_UA Class



For the functionality of the OPC-UA server an instance of this class has to be placed in a network.

On program start an own thread is created, in which the OPC-UA server runs. No further programming is necessary.

Only the existence of a configuration file (XML) is mandatory.

7.1 Interfaces

7.1.1 Servers

ClassSrv	progress for initialization (details are described below ...)	
	0	standard value at program start In this status, the "FunctSetUp" method is called. It is expected, that additional XML configuration files are provided if necessary ("AddXmlConfig" resp. "OPCUA_AddXmlConfig").
	1	status after error free execution of the method "FunctSetUp" Here the OPC-UA server is started ("OPCUA_ServerStart").
	2	status after error free execution of the method "OPCUA_ServerStart" Here the method "OPCUA_CyclicRun" is called.
ErrorCode	error codes for eventually occurring errors (details described below ...)	
	0	no errors
	-1	if the method "OPCUA_Init" has not been called as the first method
	-2	call without previous initialization - the "OPCUA_Init" method was not called
	-3	call of the method "OPCUA_AddXmlConfig" after starting the server - at this point in time, no more configuration changes are allowed
	-4	call of the method "OPCUA_ServerStart" without calling the method "OPCUA_AddXmlConfig" before
	-5	internal error when starting the OPC-UA server (see log files)
	-6	call of the method "OPCUA_CyclicRun" without calling the method "OPCUA_ServerStart" before
	-7	internal error during processing the OPC-UA protocol (see log files)
	-1001	configuration file does not exist
	-1002	length of the configuration file could not be determined
	-1003	contents of the configuration file could not be read
	-1004	reading configuration file failed (incorrect structure)

Triggers	A running counter, which is incremented with each internal cycle () and thereby returns a status as to whether or not internal processing is active.	
TraceLevel	Shows the current trace level (details described below ...) This server can also be written - so the TraceLevel can therewith be change via this server during runtime.	
	OPCUA_TRACE_LEVEL_CONTENT Data packet output (OPC UA protocol), including content	0x01
	OPCUA_TRACE_LEVEL_DEBUG ... debug information via the internal process in the OPC UA server	0x02
	OPCUA_TRACE_LEVEL_INFO ... expanded system information	0x04
	OPCUA_TRACE_LEVEL_ERROR ... serious error	0x20
	OPCUA_TRACE_LEVEL_WARNING ... system warnings	0x10
	OPCUA_TRACE_LEVEL_SYSTEM ... infrequent system events (start, stop, connect, ...)	0x08

The tracing entries are written to a file - name: **opc_log.txt**

It is located on the control in the folder **C:\sysmsg\**

The single levels are bit masks and so can be combined bitwise.

7.1.2 Clients

SigClib	object channel to SigLib (connection is established automatically)
config	Bit pattern for configuration Bit 0 IdentifierType, 0="Numeric" / 1="String" Bit 1 Reserve Bit x Reserve

7.1.3 Global Methods

Init	Initializing and creating the OPC_UA thread.
Background	The class contains a Background method, which can be used in case the class is derived. This method does not have to be activated and it has no influence on the function of the class.
FunctStart	Called once while starting the OPC UA server and signals the user that this service was started.
FunctRun	Called cyclically, if the service was started.
FunctEnd	Called once when ending the OPC UA server.
FunctSetUp	With this method, the OPC UA Server is declared the corresponding configuration files (.XML). It is called once immediately after the start. Several different configuration files can also be declared. This occurs through the function call. OUT retcode 0= OK otherwise error code (server is not started in this case!) The base implementation here loads the standard configuration file "OPC_UA.XML". So retcode complies to the return value of AddXmlConfig.
GetLasalId	reads the unique Lasal ID for a desired server IN label name of the server OUT retcode unique Lasal ID
SetValue32	sets the value of a signed 32-bit server IN lasalid unique Lasal ID IN value new value OUT retcode -1= general error 0= access denied 1= OK
SetValueU32	Equivalent to the "Setvalue32" method for the data type UDINT
SetValueF32	Equivalent to the "Setvalue32" method for the data type REAL
GetValue32	Reads the value of a signed 32-bit server

	IN pvalue — the read value is written to this address IN lasalid — unique Lasal ID OUT retcode — -1= general error 1= OK
GetValueU32	Equivalent to the “Getvalue32" method for the data type UDINT
GetValueF32	Equivalent to the “Getvalue32" method for the data type REAL
GetString16Crc	Returns the CRC value for the transferred Lasal ID IN lasalid — unique Lasal ID OUT retcode — CRC value
GetString16	Reads the contents of a server of the type STRING IN pdst — pointer to which the value should be written IN max_chrlength — maximum length of the string IN lasalid — unique Lasal ID OUT retcode — -1= general error 1= OK
SetString16	Sets the value of a server of the type STRING IN lasalid — unique Lasal ID IN pstr — pointer to the new value OUT retcode — -1= general error 0= access denied 1= OK
CB_activateDS **CB_prepaireDS**	The Callback (CB) “CB_activateDS” is called, if a client wants to transfer and activate a data set to the control. This callback is used in the control program as a trigger for reading and activating the desired settings data set. The "f_CB_ prepaireDS" callback is called when a client requests a data set. This callback is used in the control program as a trigger for providing desired settings data set. IN pID — unique ID of the data set IN pName — name of the data set IN pPath — path, where the data set is located OUT retcode — 0= OK otherwise error code
CB_alarmList	The OPC UA server calls this function during initialization. With this method, the OPC UA server requests the list of all active alarms. OUT retcodereserved for future tasks, is not evaluated
CB_fileSystem	This Callback is called, if a file changed. IN typ — type of the file change (1= file new, 2= file deleted, 3 = file changed) IN pPath — path incl. file name and extension of the file OUT retcode — reserved for future tasks, is not evaluated

RemoteReadStatistic	This method is called after each successful reading process on a remote server. IN description — describes the read remote server IN count — number of variables read without any errors OUT retcode — reserved for future tasks, is not evaluated
RemoteWriteStatistic	This method is called after each successful writing process on a remote server. IN description — describes the written remote server IN count — number of variables written without any errors OUT retcode — reserved for future tasks, is not evaluated
RemoteReadError	This method is called for each single variable, after its reading process on a remote server failed. IN description — describes the read remote server IN node — describes the single variable on the remote server IN status — OPC-UA error code OUT retcode — reserved for future tasks, is not evaluated
RemoteWriteError	This method is called for each single variable, after its writing process on a remote server failed. IN description — describes the read remote server IN node — describes the single variable on the remote server IN status — OPC-UA error code OUT retcode — reserved for future tasks, is not evaluated
CurrentExternalServerStatus	This method is called with each status change of a remote server. IN Id — Id of the remote server IN Url — Url of the remote server IN Endpoint — end point of the remote server IN status — 0=OpcUa_Good 0x808A0000=OpcUa_BadNotConnected OUT retcode — reserved for future tasks, is not evaluated
SetParameter	can be used to set process specific parameters IN ParaNr — parameter number IN Value — value to be set OUT retcode — 0 for success -1 if the parameter could not be set valid parameter: OPC_UA_PAR_SET_DELAYTIME ... delay for the OPC_UA thread
NewSystemTime	The method is called if an OPC-UA clients tries to set a new system time. If the user wants to evaluate the system time, it can be overwritten. If this method is not overwritten, it calls the private SetSystemTime method and assumes the system time automatically.

SetTimeZoneOffset	This method can be used for setting a time zone offset or a Summer Time offset. This is necessary for calculating a valid UNIX time stamp from the current system time. (The UNIX time stamp must always refer to UTC by definition.)
OpcUaThread	OPC-UA services are processed in this thread. IN pthis — pointer to the own instance

7.1.4 Private Methods

OPC_UA	Constructor initializes the OPC-UA interface OUT ret_code — ConfStates (ask SIGMATEK developer for more information)
AddXmlConfig	reads a new/additional configuration file. So additional elements in the OPC-UA address space are registered. IN dpne — path + file name + ext. where the data set is located OUT retcode — 0= OK otherwise negative error code
SetValue	Function executing the internal writing procedure of a server (see SetValue32, SetValueF32, …) IN lasalid — unique Lasal ID IN value — new value OUT retcode — -1= general error 0= access denied 1= OK
GetValue	function executing the internal reading procedure of a server (see GetValue32, GetValueF32, …) IN pvalue — pointer to the read value IN lasalid — unique Lasal ID OUT retcode — -1= general error 1= OK
AllActiveAlarms	With this method, the list of active alarms can be sent to the OPC UA server. This method must be called only once during the program start to initialize the list of active alarms. Together will all other alarm related functions, this call has to be executed threadsafe. IN alarmList — pointer to the list of active alarms IN listCount — number of alarms in the list OUT state — 0 Via the "GetAllActiveAlarms" function, OPC UA clients can query the list of active alarms.

AlarmChanged	With this method, a change in an alarm can be sent to the OPC UA server. Each change of an alarm has to be reported. Both activation and deactivation of an alarm. With this method, the list of the current alarms is kept up to date. Together will all other alarm related functions, this call has to be executed threadsafe. IN alarm — information about the changed alarm OUT state — 0
SetvalueF32Changed	Equivalent to the "Setvalue32Changed" method for the data type "REAL".
SetvalueU32Changed	Equivalent to the "Setvalue32Changed" method for the data type "UDINT".
Setvalue32Changed	With this method, changes (of type DINT) to the settings data can be sent to the OPC UA server. Together will all other “Setvalue..Changes” functions, this call has to be executed threadsafe. IN change — general properties of the parameter change IN oldValue — value before the change IN newValue — current value / value after the change OUT state — 0 Using OPC UA Event, OPC UA clients can be informed of changes in the settings data.
DatasetActivationFinished	With this method, the OPC UA server can be informed when settings data is activated. IN status — status of the activation (0: error free, !=0: error code) IN datasetId — unique Id of the transfer / data set IN datasetName — name of the settings data set IN path — path where the according file was saved OUT state — 0 Using OPC UA Event, OPC UA clients can be informed when settings data is activated.
DatasetPreparationFinished	With this method, the OPC UA server can be informed that the preparation of a data set for transmission to a client has been completed. IN status — status of the activation (0: error free, !=0: error code) IN datasetId — unique Id of the transfer / data set IN datasetName — name of the settings data set IN path — path where the according file was saved OUT state — 0 Using OPC UA Event, OPC UA clients can be informed when settings data has been prepared. A client can then read the settings data via the “DownloadFile”, method from the OPC UA server.
InitDatasetWorkingPath	With this method, the default path for operations with settings data for runtime can be defined. This path is, in addition to paths from the configuration, valid for all file operations. If this path is set, it is used as the default path for file operations

	without path specifications. E.g.: If the DatasetWorkingPath was set to c:\datenset\, the file is stored in the c:\datensatz\text.txt“ directory when UploadFile is called with the "test.txt" parameter. IN path default path specification OUT retcode 0
InitAlarmCallback	With this method, the OPC UA server can provided with a callback function. The OPC UA server calls this function during initialization. With this method, the OPC UA server requests the list of all active alarms. IN f_CB_alarmList pointer to the Callback function OUT retcode 0
InitDatasetCallback	With this method, the OPC UA server can be provided with two callback functions. IN f_CB_activateDS CallBack for activating settings data sets f_CB_prepaireDS Callback for providing settings data sets OUT retcode 0 The Callback (CB) "f_CB_activateDS” is called, if a client wants to transfer and activate a data set to the control. This callback is used in the control program as a trigger for reading and activating the desired settings data set. The "f_CB_ prepaireDS" callback is called when a client requests a data set. This callback is used in the control program as a trigger for providing desired settings data set.
InitVersionId	With this method, the OPC UA server can sent a unique ID (version number). This ID can be later used for customer and control-specific implementations. IN versionId unique identification of the control version OUT retcode 0
SetTraceLevel	With this method, the TraceLevel can be changed during runtime. IN traceLevel TraceLevel to be used OUT retcode 0
InitFileSystemCallback	With this method, the OPC UA server can provided with a callback function. IN f_CB_fileSystem Callback for FileSystem changes OUT retcode 0 The "f_CB_fileSystem" callback is called when a client triggers a change in the file system with a function call. All file functions (Upload File, Download File, Activate Dataset, Prepare Dataset) for example, thereby trigger changes in the file system and subsequently call this callback function.
SetOptions	With this function option can be set that change the program sequence. IN options USE:HOSTNAME-AS-BROWSENAME USE:ALPHANUMERIC-IDENTIFIERS OUT retcode 0

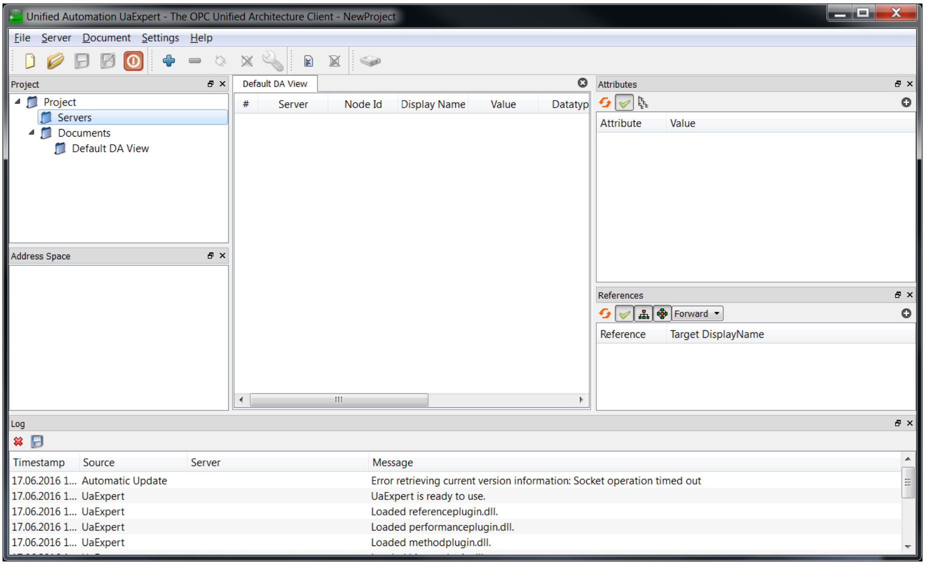
AlarmChangedUC	Complies to the method "AlarmChanged". The difference is that strings in this method are transferred as an array of 16-bit values. So any UniCode characters can be transferred. Together will all other alarm related functions, this call has to be executed threadsafe. IN alarm — information about the changed alarm OUT retcode — 0
SetvalueStringChanged	Equivalent to the "Setvalue32Changed" method for the data type "CHAR". Together will all other “Setvalue..Changes” functions, this call has to be executed threadsafe. IN change — general properties of the parameter change IN oldValue — value before the change IN newValue — current value / value after the change OUT state — 0
SetvalueStringChangedUC	Complies to the method "SetvalueStringChanged". The difference is that the input parameter in this method is transferred as an array of 16-bit values. So any UniCode characters can be transferred. Together will all other “Setvalue..Changes” functions, this call has to be executed threadsafe.

8 Win Program UaExpert

This program is not from SIGMATEK and not necessary for the real operation of an OPC-UA communication.

But this tool can help in first commissioning.

8.1 Setup Connection

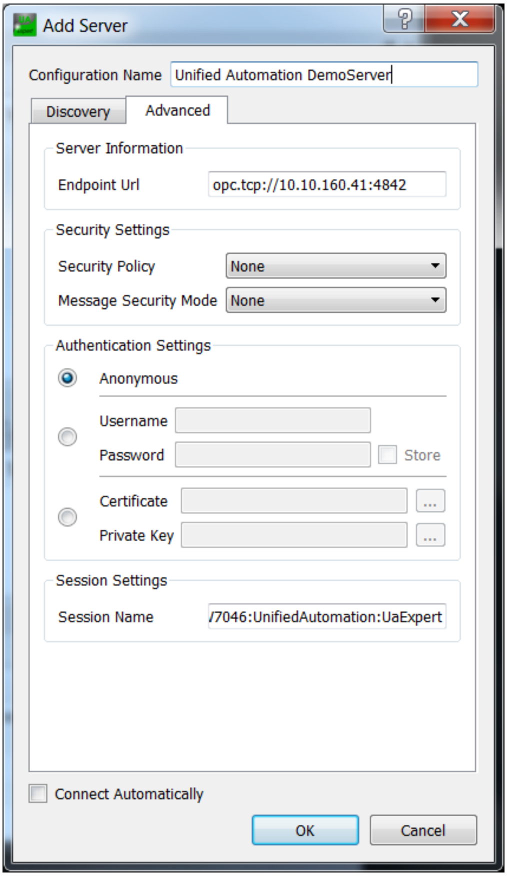

Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject
File Server Document Settings Help
Project
Project
Servers
Documents
Default DA View
Default DA View
# Server Node Id Display Name Value Datatyp
Attributes
Attribute Value
Address Space
References
Forward
Reference Target DisplayName
Log
Timestamp Source Server Message

```
17.06.2016 1... Automatic Update    Error retrieving current version information: Socket operation timed out
17.06.2016 1... UaExpert            UaExpert is ready to use.
17.06.2016 1... UaExpert            Loaded referenceplugin.dll.
17.06.2016 1... UaExpert            Loaded performanceplugin.dll.
17.06.2016 1... UaExpert            Loaded methodplugin.dll.
```

After the first start no project exists.

- Right click on “servers” in the section Project - click on “Add...”

continue on the next page ...



Configuration Name: Unified Automation DemoServer

Server Information
Endpoint Url: opc.tcp://10.10.160.41:4842

Security Settings
Security Policy: None
Message Security Mode: None

Authentication Settings
[✓ Anonymous]
Username
Password
Store
Certificate
Private Key

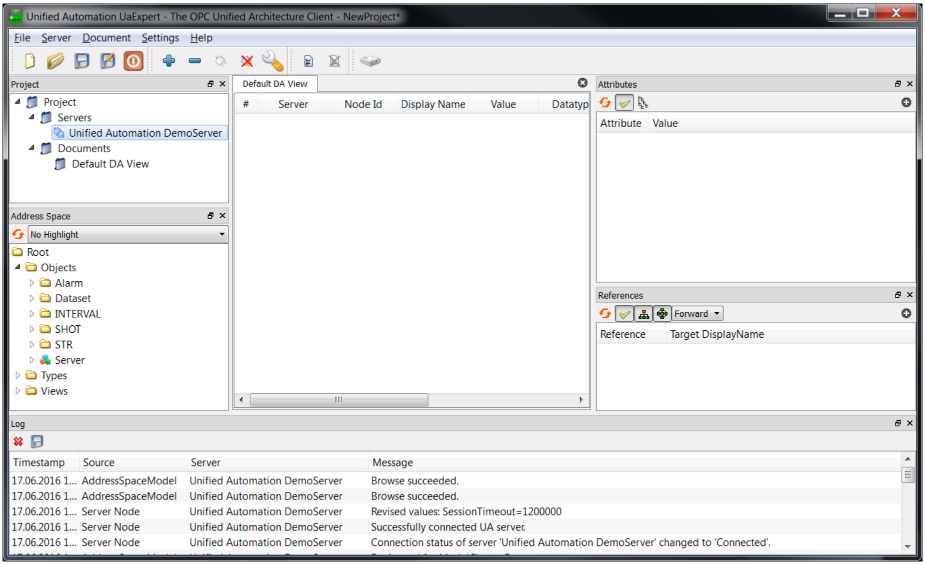
Session Settings
Session Name: /7046:UnifiedAutomation:UaExpert

Connect Automatically

- here the entpoint URL has to be entered

`opc.tcp://10.10.160.41:4842`

…	the IP address is that of the PLC
…	the port is 4842 for SIGMATEK


Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject*
File Server Document Settings Help
Project
Project
Servers
Unified Automation DemoServer
Documents
Default DA View
Address Space
No Highlight
Root
Objects
Alarm
Dataset
INTERVAL
SHOT
STR
Server
Types
Views
Default DA View
# Server Node Id Display Name Value Datatyp
Attributes
Attribute Value
References
Forward
Reference Target DisplayName
Log

Timestamp	Source	Server	Message
17.06.2016 1...	AddressSpaceModel	Unified Automation DemoServer	Browse succeeded.
17.06.2016 1...	AddressSpaceModel	Unified Automation DemoServer	Browse succeeded.
17.06.2016 1...	Server Node	Unified Automation DemoServer	Revised values: SessionTimeout=1200000
17.06.2016 1...	Server Node	Unified Automation DemoServer	Successfully connected UA server.
17.06.2016 1...	Server Node	Unified Automation DemoServer	Connection status of server 'Unified Automation DemoServer' changed to 'Connected'.

- with a right click on the server and “Connect” the connection is established.

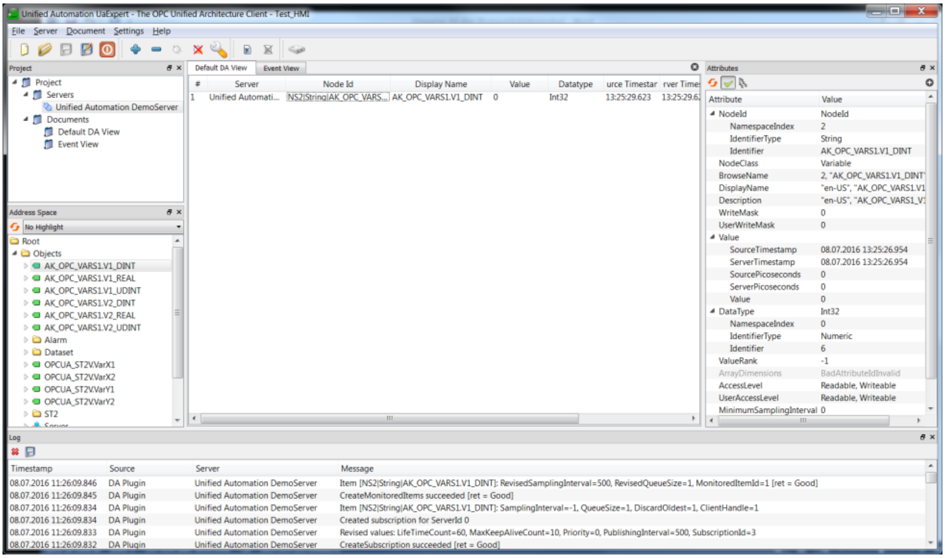
- now automatically the OPC-UA client requests the variable list from the OPC server and displays it here.

The OPC server generates this variable list according to its own OPC XML file.

8.2 Data exchange

The normal data exchange in SIGMATEK systems is realized with server variables.

Supported at the moment: DINT, UDINT, REAL and STRING



#	Server	Node Id	Display Name	Value	Datatype	urce Timestar	rver Time
1	Unified Automati...	NS2|String|AK_OPC_VARS...	AK_OPC_VARS1.V1_DINT	0	Int32	13:25:29.623	13:25:29.6

Attribute	Value
NodeId	NodeId
NamespaceIndex	2
IdentifierType	String
Identifier	AK_OPC_VARS1.V1_DINT
NodeClass	Variable
BrowseName	2, "AK_OPC_VARS1.V1_DINT"
DisplayName	"en-US", "AK_OPC_VARS1.V1
Description	"en-US", "AK_OPC_VARS1_V1
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	08.07.2016 13:25:26.954
ServerTimestamp	08.07.2016 13:25:26.954
SourcePicoseconds	0
ServerPicoseconds	0
Value	0
DataType	Int32
NamespaceIndex	0
IdentifierType	Numeric
Identifier	6
ValueRank	-1
ArrayDimensions	BadAttributeIdInvalid
AccessLevel	Readable, Writeable
UserAccessLevel	Readable, Writeable
MinimumSamplingInterval	0

```
Timestamp                Source     Server                          Message
08.07.2016 11:26:09.846  DA Plugin  Unified Automation DemoServer   Item [NS2|String|AK_OPC_VARS1.V1_DINT]: RevisedSamplingInterval=500, RevisedQueueSize=1, MonitoredItemId=1 [ret = Good]
08.07.2016 11:26:09.845  DA Plugin  Unified Automation DemoServer   CreateMonitoredItems succeeded [ret = Good]
08.07.2016 11:26:09.834  DA Plugin  Unified Automation DemoServer   Item [NS2|String|AK_OPC_VARS1.V1_DINT]: SamplingInterval=-1, QueueSize=1, DiscardOldest=1, ClientHandle=1
08.07.2016 11:26:09.834  DA Plugin  Unified Automation DemoServer   Created subscription for ServerId 0
08.07.2016 11:26:09.833  DA Plugin  Unified Automation DemoServer   Revised values: LifeTimeCount=60, MaxKeepAliveCount=10, Priority=0, PublishingInterval=500, SubscriptionId=3
08.07.2016 11:26:09.832  DA Plugin  Unified Automation DemoServer   CreateSubscription succeeded [ret = Good]
```

- in the area “AddressSpace” on the left side the variable list read from the PLC can be found
- the value of the according variable can be found to the right under “Value”
- updating here is only done when selecting
- with Drag & Drop a variable can also be moved to the center view field
- variables placed here are updated cyclically
- the values can be changed

8.3 Alarms

For alarms OPC-UA provides a flexible alarm handling.

The base class cannot access the SIGMATEK alarms.

For this SIGMATEK provides an expanded OPC-UA class.

8.4 Events

For alarms OPC-UA provides a flexible event handling.

The base class cannot access the SIGMATEK events.

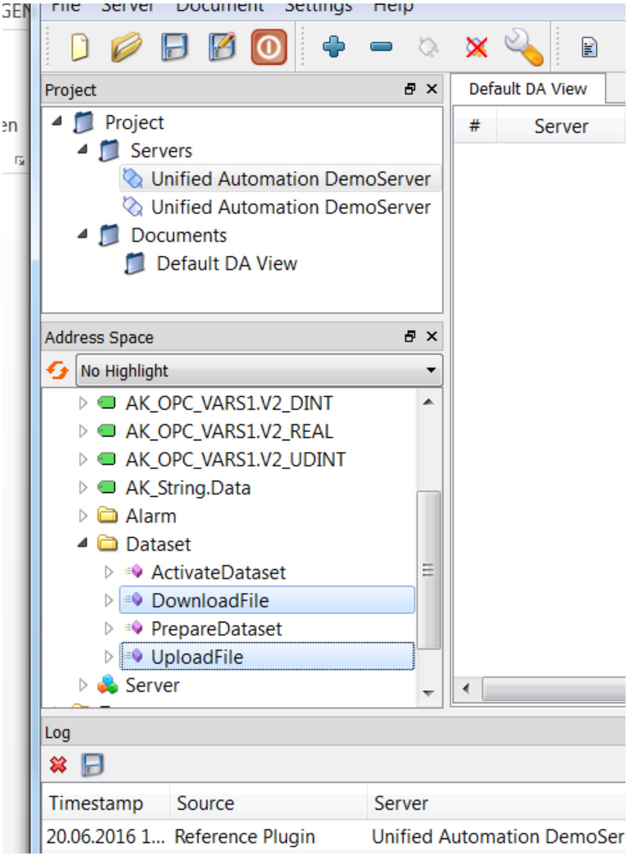
An expansion to also support this functionality is already planned.

8.5 File Transfer

A file transfer can be executed in both directions. But in both cases the client writes.

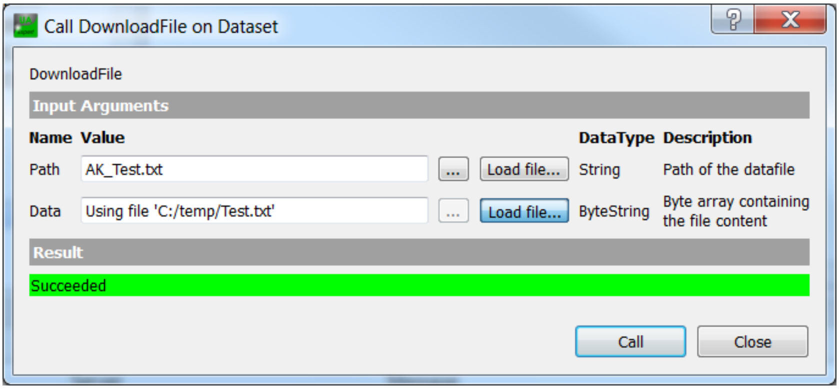
In the base class the server has no influence on the time of the transfer.

In the program UaExpert, these functions can be found on the left side in the section “Address Space” in the entry “Dataset”.



Timestamp	Source	Server
20.06.2016 1...	Reference Plugin	Unified Automation DemoSer

8.5.1 File from Client (Win) to Server (PLC)


Call DownloadFile on Dataset

DownloadFile

Input Arguments

Name	Value			DataType	Description
Path	AK_Test.txt	...	Load file...	String	Path of the datafile
Data	Using file 'C:/temp/Test.txt'	...	Load file...	ByteString	Byte array containing the file content

Result

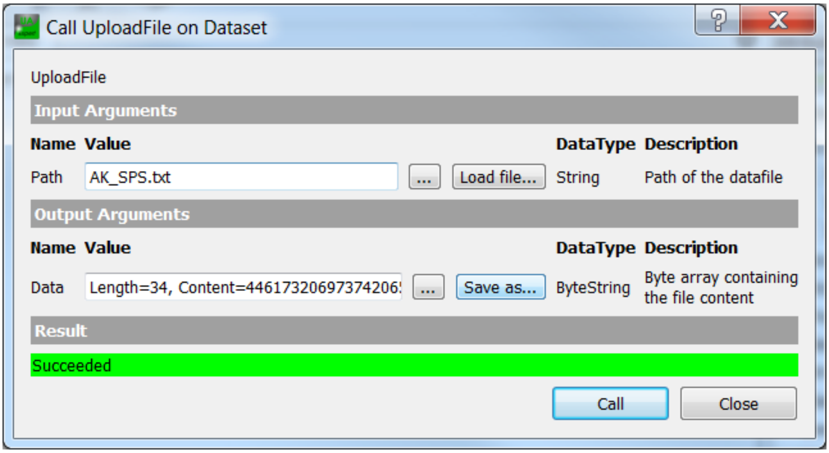
Succeeded

"Path"	defines the target path on the PLC including the file name. This path has to be entered in the PLC XML file in the section Release! Default "C:\OPCUA\" Without path definition the first DIR entered in the XML is used.
Data	defines the source on the client.

An error indicates, that e.g. the OPC-UA DIR is missing on the PLC.

It is only possible to write to this DIR!

8.5.2 File from Server (PLC) to Client (Win)


Call UploadFile on Dataset

UploadFile

Input Arguments

Name	Value	DataType	Description
Path	AK_SPS.txt	String	Path of the datafile

Output Arguments

Name	Value	DataType	Description
Data	Length=34, Content=44617320697374206!	ByteString	Byte array containing the file content

Result

Succeeded

"Path"	defines the source path on the PLC including the file name. This path has to be entered in the PLC XML file in the section Release! Default "C:\OPCUA\" Without path definition the first DIR entered in the XML is used.
Data	is written after pressing the Cal button and then can be saved in a file with “Save as”.

An error indicates, that e.g. the OPC-UA DIR is missing on the PLC.

It is only possible to read from this DIR!

Documentation Changes

Change date	Affected page(s)	Chapter	Note
13.02.2015	2	1.1 Delivery 1.2 Placement	Delivery and placement added
26.06.2015	1	2 … configuration	Description of Trace / TraceLevel
26.06.2015	3	3.2 Servers	Description of the class servers (ClassSvr, ErrorCode, Trigger, TraceLevel)
27.06.2015	1	3.1 Functions	AllActiveAlarms
27.06.2015	1	3.1 Functions	AlarmChanged
27.06.2015	1	3.1 Functions	Setvalue32Changed, …
27.06.2015	1	3.1 Functions	DatasetActivationFinished
27.06.2015	1	3.1 Functions	DatasetPreparationFinished
27.06.2015	1	3.1 Functions	InitDatasetWorkingPath
27.06.2015	1	3.1 Functions	InitAlarmCallback
27.06.2015	1	3.1 Functions	InitDatasetCallback
27.06.2015	1	3.1 Functions	InitVersionId
27.06.2015	1	3.1 Functions	SetTraceLevel
27.06.2015	1	3.1 Functions	InitFileSystemCallback
27.06.2015	1	3.1 Functions	CB_alarmList
28.06.2015	1	1.3 Supported OPC-UA Services	Adapted
28.06.2015	1	1.4 Supported OBP-UA Features and Profiles	Adapted
26.06.2016	3	1.3 Supported OPC-UA Services	Adapted formatting
26.06.2016	5-8	2.0 Configuration with XML File	To expand configuration possibilities for the access to external OPC-UA servers
26.06.2016	18-19	4.5 Data Synchronization between Different OPC-UA Servers	Expanded with this new function
xx.08.2016	all	all	Documentation updated and expanded
21.10.2016	25, 27, 28	7.1.3 Global Methods	Added methods