

SIGMATEK

OPC-UA Server

Herausgeber: SIGMATEK GmbH & Co KG
A-5112 Lamprechtshausen
Tel.: +43/6274/4321
Fax: +43/6274/4321-18
Email: office@sigmatek.at
WWW.SIGMATEK-AUTOMATION.COM

Copyright © 2016
SIGMATEK GmbH & Co KG

Originalsprache

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder in einem anderen Verfahren) ohne ausdrückliche Genehmigung reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhaltliche Änderungen behalten wir uns ohne Ankündigung vor. Die SIGMATEK GmbH & Co KG haftet nicht für technische oder drucktechnische Fehler in diesem Handbuch und übernimmt keine Haftung für Schäden, die auf die Nutzung dieses Handbuches zurückzuführen sind.

Inhalt

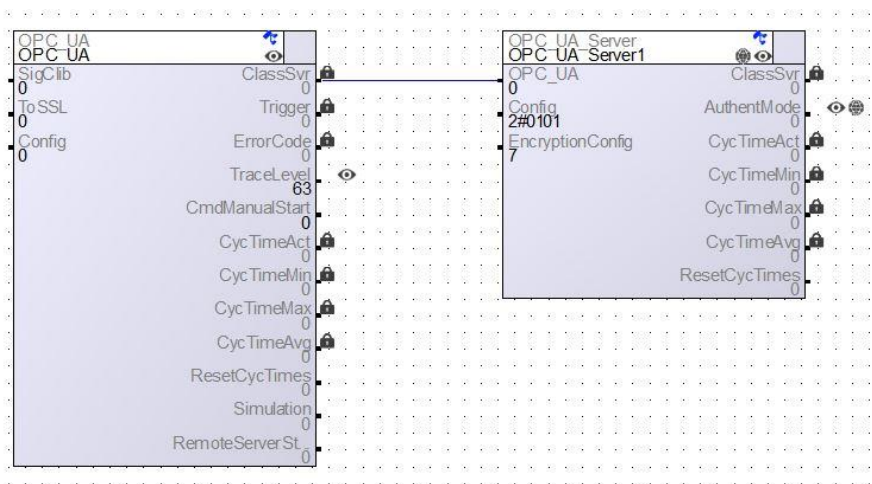
1	OPC-UA Server Einführung	4
2	Zugriffsberechtigung	5
3	Verbindungsaufbau und Limits.....	6
3.1	Sessions	6
3.2	Subscriptions	6
3.3	MonitoredItems	6
4	SSL Verschlüsselung.....	7
4.1	Allgemeine Informationen.....	7
4.2	Verwaltung von Zertifikaten.....	8
4.3	Sicherheit der OPC-UA-Verbindung	8
4.4	SPS-spezifische Hinweise	9
4.5	SSL-Einstellungen am Client.....	10
5	Einfacher Datenaustausch	11
5.1	Prinzip für einfache Daten	11
5.2	Implementierung ins SPS-Projekt.....	12
6	Konfiguration mittels XML-Datei.....	14
6.1	Grundsätzlicher Aufbau	14
6.2	Beispiel für OPC-UA-Variablen mit 2 Konfigurationsdateien.	16
6.3	Übertragung komplexer Daten: ByteString.....	18
6.3.1	ByteStrings konstanter Länge.....	18

- 6.3.2 ByteStrings variabler Länge..... 19
- 6.4 Übertragung komplexer Daten: Strukturierte Variable 20
- 7 OPC-UA Funktionen.....22
 - 7.1 Lesen/Schreiben von Variablen 22
 - 7.2 MonitoredItems 22
 - 7.3 Datei Download 23
 - 7.4 Datei Upload 23
- 8 Klasse OPC-UA_Server24
 - 8.1 Schnittstellen 25
 - 8.1.1 Server 25
 - 8.1.2 Clients..... 26
 - 8.1.3 Globale Methoden 27
 - 8.1.4 Private Methoden 34
 - 8.2 FAQ zu Leistungsdaten und Speicherbedarf..... 37
- 9 Win-Programm UaExpert38
 - 9.1 Verbindung einrichten..... 38
 - 9.2 Verbindung aufbauen..... 40
 - 9.3 Datenaustausch 42
 - 9.4 Alarme..... 43
 - 9.5 Events 43
 - 9.6 File-Transfer 44
 - 9.6.1 File von Client (Win) auf Server (SPS) 45
 - 9.6.2 File von Server (SPS) zu Client (Win) 46

10 Anhang A 47

1 OPC-UA Server Einführung

Für die OPC-UA Server Funktionalität muss die OPC-UA Klasse importiert und in einem Netzwerk platziert werden. Weiters ist eine OPC-UA_Server-Klasse zu platzieren und an die OPC-UA-Klasse anzuschließen.



Wichtige Hinweise findet man auch in der Dokumentation zur Klasse `OPC-UA`.

2 Zugriffsberechtigung

Es ist eine einfache Benutzerverwaltung samt Zugriffskontrolle integriert.
Ab Version 3.5 unterstützt der OPC-UA-Server auch optionale Userrollen.

Dazu wird eine XML-Datei namens "**UserConfiguration.xml**" herangezogen, welches sich im Ordner "**C:\OPC-UA**" befinden muss. Fehlt dieser Ordner, so muss die Datei in der Root des Laufwerkes C auf der Steuerung liegen.

- ist diese Datei **nicht** vorhanden, so findet **keine** Überprüfung statt und es sind nur anonyme Anmeldungen möglich. Der Client darf also in diesem Falle den Connect nur anonym ohne Benutzername und ohne Passwort durchführen.
- ist diese Datei vorhanden, so wird vom OPC-UA-Server automatisch bei jedem clientseitigen Verbindungsaufbau überprüft, ob die mit dem Connect erhaltene Kombination von Benutzername und Passwort im File hinterlegt ist. Dabei ist unbedingt auf Gross-/Kleinschreibung zu achten.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
  <Users>
    <User Name="BertiBediener" Password="x123" Userrole="5" />
    <User Name="EmilExperte" Password="y345" Userrole="-1" />
    <User Name="RosiRatlos" Password="z987" Userrole="0" />
  </Users>
</Config>
```

- Mittels des Klassen-Servers "AuthentMode" kann die Anmeldung auf anonym geschaltet werden. Somit kann z.B. vorübergehend die Authentifizierung außer Kraft gesetzt werden, indem man den Klassen-Server "AuthentMode" auf #1 setzt.
- Das optionale Attribut "Userrole" ist als Bitmaske zu interpretieren, es kann für bis zu 32 Schreib-Berechtigungen genutzt werden.
Beim Schreib-Zugriff auf einen Node wird geprüft, ob bei jenem Node ebenfalls eine "Userrole" angegeben wurde – ist dies der Fall, so muss mindestens eines der 32 möglichen Bits übereinstimmen für die Schreib-Erlaubnis.
Userrole="0" ... der Benutzer hat keinerlei Schreibrechte
Userrole="-1" ... der Benutzer hat alle Schreibrechte
Userrole="x" ... x = Wert für beliebige Bitmaske (5 = Rolle 1 und Rolle 3)

3 Verbindungsaufbau und Limits

Der Server stellt sogenannte Endpoints zum Verbindungsaufbau zur Verfügung. Für einen Verbindungsaufbau wird ein Endpoint ausgewählt, der die entsprechenden Verschlüsselungsanforderungen erfüllt. Stellt der Server keinen entsprechenden Endpoint zur Verfügung, müssen die Verschlüsselungseinstellungen entsprechend angepasst werden.

Die Endpoints, die ein Server zur Verfügung stellt, können unter Beachtung des Betriebssystems konfiguriert werden. Siehe hierzu **Fehler! Verweisquelle konnte nicht gefunden werden. (Fehler! Verweisquelle konnte nicht gefunden werden.)**.

Die hier angegebenen Grenzwerte sind diejenigen, die durch die Software vorgegeben sind. Es ist jedoch zusätzlich darauf zu achten, welche Applikation auf welcher Hardware läuft. Dies beeinflusst die Antwortzeiten des Servers und es ist darauf zu achten, dass ggf. applikationsspezifisch kleinere Werte verwendet werden sollten.

3.1 Sessions

Nach der Eröffnung einer Verbindung, wird vom Client eine Session geöffnet. Die Anzahl an gleichzeitigen Sessions am Server ist begrenzt. Ein Client kann hierbei höchstens 25 Sessions öffnen und die Gesamtzahl der Sessions am Server beträgt 50. Ist die maximale Anzahl an Sessions am Server geöffnet, werden weitere Versuche mit dem Fehler 0x80560000 (BadTooManySessions) abgewiesen.

3.2 Subscriptions

Pro Session ist die Anzahl der möglichen Subscriptions auf 20 begrenzt. Weitere Versuche eine Subscription anzulegen führen zum Fehler 0x80770000 (BadTooManySubscriptions).

3.3 MonitoredItems

Die maximale Anzahl an MonitoredItems pro Subscription beträgt 1000. Darüber hinaus MonitoredItems anzulegen führt zum Fehler 0x80DB0000 (BadTooManyMonitoredItems).

4 SSL Verschlüsselung

4.1 Allgemeine Informationen

Wird Verschlüsselung auf einer SPS grundsätzlich unterstützt (siehe Kapitel 3.3), wird beim Start von OPC-UA pro Schnittstelle ein Zertifikat angelegt, falls diese nicht vorhanden sind. Die Zertifikate befinden sich im Ordner `C:\opc_ua` auf der SPS. Für die erste Schnittstelle werden zwei Dateien mit Namen `cert.der` und `key.pem` erstellt. Die Dateien für weitere Schnittstellen heißen danach `cert<num>.der` und `key<num>.pem`, wobei `num` bei 1 beginnt und fortlaufend erhöht wird. Es ist jedoch nicht zwingend erforderlich, dass alle Nummer von 1 weg vergeben werden.

Die Datei mit der Endung *pem* enthält den privaten Schlüssel für das Zertifikat. Die Datei mit der Endung *der* beinhaltet das Zertifikat. Die Zertifikate müssen dem X.509 v3 Standard genügen, um verwendet werden zu können.

Die Zertifikate können jedoch auch vom Administrator zur Verfügung gestellt werden und werden, wie oben beschrieben, auf der Steuerung abgelegt.

Zertifikate beinhalten Informationen, ab wann und bis wann diese gültig sind.

Achtung: Es ist darauf zu achten, dass beim Erstellen des Zertifikats die Uhrzeit der SPS korrekt eingestellt ist.

In beiden Fällen, wenn ein Zertifikat noch nicht oder nicht mehr gültig ist, wird dieses von der Gegenstelle abgelehnt. Das ungültige Zertifikat muss gelöscht werden und durch ein neues ersetzt, oder neu generiert werden.

Achtung: Es ist darauf zu achten, dass der private Schlüssel Dritten nicht zugänglich ist, da sonst der verschlüsselte Datenverkehr mitgelesen werden kann.

Hinweis: Aktuell können Zertifikate lediglich zum Zwecke der Verbindungssicherheit verwendet werden. Es ist nicht möglich, auf der Steuerung ein signiertes Zertifikat zu erstellen oder zu verwenden. Ebenso gibt es keine Möglichkeit, Verkettete Zertifikate (CA-Zertifizierung) zu verwenden.

Hinweis: Aktuell können keine Zertifikate zur Authentifizierung verwendet werden, sondern lediglich Benutzername und Passwort.

Hinweis: Soll ein Zertifikat über die Methode `CreateApplCertificate` (siehe auch Klassendokumentation der `OPC-UA`-Klasse) erstellt werden, ist es notwendig, Daten der Zertifikatsausstellers anzugeben. Dabei ist zu beachten, dass das Land als 2-Buchstabencode (zum Beispiel „AT“ für Österreich) angegeben werden muss. Andernfalls quittiert `OPC-UA` den Aufruf mit `BadInvalidArgument`.

4.2 Verwaltung von Zertifikaten

Zur Verwaltung und Anzeige von Zertifikaten auf der Steuerung, steht das AddOn **SSLCertificate** zur Verfügung. Dieses AddOn ermöglicht es, die Zertifikate zwischen den einzelnen Ordner zu verschieben und diese zu löschen.

Hinweis: Eine Applikation sollte die Möglichkeit vorsehen, Zertifikate löschen, verschieben und erstellen/hinunterladen zu können, da nach Ablauf eines Zertifikats keine Sichere Verbindung über OPC-UA mehr möglich ist.

4.3 Sicherheit der OPC-UA-Verbindung

Ab der Version V.3.0 des OPC-UA-Servers ist die SSL-Verschlüsselung implementiert. Folgende SecurityPolicies werden aktuell unterstützt:

- None
- Basic256¹
- Basic256Sha256
- Aes128_Sha256_RsaOaep

Dabei können die Sicherheitsmodi

- None: Keine Verschlüsselung
- Sign: Nachricht wird nur signiert
- SignAndEncrypt: Nachricht wird verschlüsselt und signiert

verwendet werden.

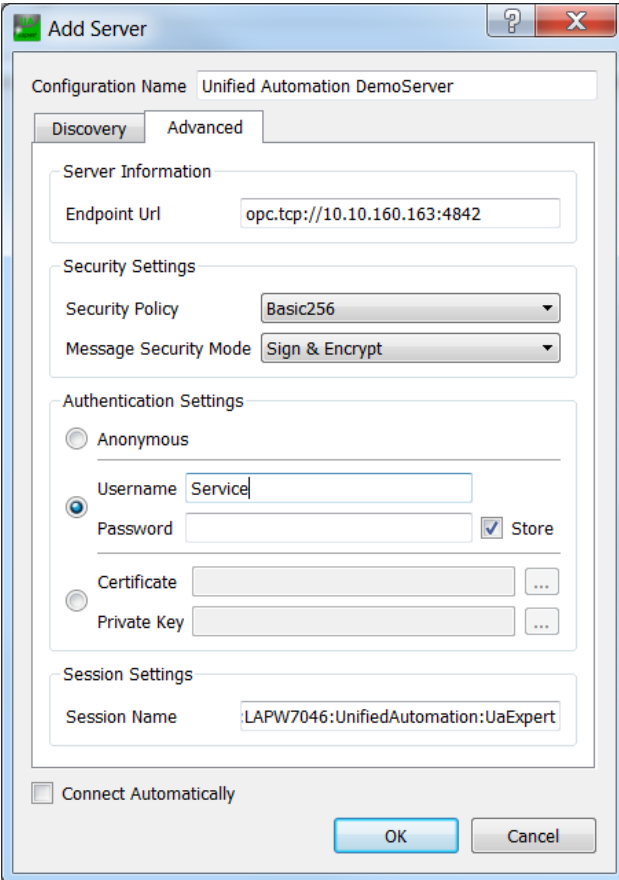
¹ Diese SecurityPolicy wird als nicht mehr sicher erachtet und sollte daher nicht mehr verwendet werden.

4.4 SPS-spezifische Hinweise

- Die SSL-Verschlüsselung wird **unter RTOS-OS NICHT unterstützt**
- Bei Salamander-OS wird ab der Version 09.03.116 SSL unterstützt
- Die für die Zertifikate notwendige Ordnerstruktur wird beim Client-SSL-Connect automatisch innerhalb des OPC_UA-Ordners angelegt
- Ein neues Zertifikat kommt vorerst automatisch in den Ordner "rejected" (steht für "abgewiesen")
- Ein akzeptiertes Zertifikat muss in Ordner "trusted" (steht für "vertrauenswürdig") verschoben werden.
Solange das Zertifikat nicht in diesem Ordner liegt, kann der dazugehörige Client keine Verbindung mit dem OPC_UA-Server aufbauen.
- Soll ein Zertifikat später zurückgezogen werden, so muss dieses in den Ordner "revoked" (steht für "widerrufen") verschoben werden
- Seit Juli 2018 bietet SIGMATEK ein AddOn für die Verwaltung von SSL-Zertifikaten
- Vor dem Erstellen der Zertifikate ist darauf zu achten, dass die Uhrzeit der SPS richtig eingestellt ist.
- Die SecurityPolicys Basic256Sha256 und Aes128_Sha256_RsaOaep werden ab Salamander-OS Version 09.03.160 unterstützt.

4.5 SSL-Einstellungen am Client

Als Beispiel dient der UA-Expert



Add Server

Configuration Name: Unified Automation DemoServer

Discovery | **Advanced**

Server Information

Endpoint Url: opc.tcp://10.10.160.163:4842

Security Settings

Security Policy: Basic256

Message Security Mode: Sign & Encrypt

Authentication Settings

☐ Anonymous

☒ Username: Service

Password: ☒ Store

☐ Certificate: ...

☐ Private Key: ...

Session Settings

Session Name: LAPW7046:UnifiedAutomation:UaExpert

☐ Connect Automatically

OK Cancel

5 Einfacher Datenaustausch

In diesem Abschnitt wird der Austausch von Basisdatentypen behandelt.

Der normale Datenaustausch geschieht bei SIGMATEK-Systemen über Server-Variablen von einer oder mehreren Klassen.

Unterstützt werden momentan einfache Daten:
DINT, UDINT, REAL und STRING

sowie komplexe Daten:
BYTESTRING und Strukturen

Der Zugriff auf den SIGMATEK OPC-UA-Server geschieht über dessen Endpoint-Url.

Zur Kommunikation wird der Port 4842 verwendet.

Beispiel für Endpoint-Url: `opc.tcp://10.10.160.41:4842`

5.1 Prinzip für einfache Daten

- Der Anwender muss alle OPC-UA-Variablen im SPS-Projekt als solche deklarieren. Es ist wohl sinnvoll, für die OPC-UA-Variablen eigene Klassen zu erstellen und diese in einem Netzwerk zu platzieren. Siehe nachfolgende Erläuterungen zur Implementierung.
- Alle zu übertragenden Variablen werden in ein XML-File eingetragen. Dies übernimmt das SIGMATEK-System völlig selbstständig und vollautomatisch! Das XML-File wird beim Download des Projekts in die CPU übertragen.
- Beim Hochlauf der SPS liest und interpretiert der OPC-UA-Server das XML-File und generiert daraus den OPC-UA-Adressraum für den einfachen Datenaustausch.
- Der OPC-UA Server steht für die Außenwelt zur Verfügung, somit kann sich ein OPC-UA-Client mit dem Server verbinden und sämtliche OPC-UA-Variablen und deren Eigenschaften lesen bzw. beschreiben.

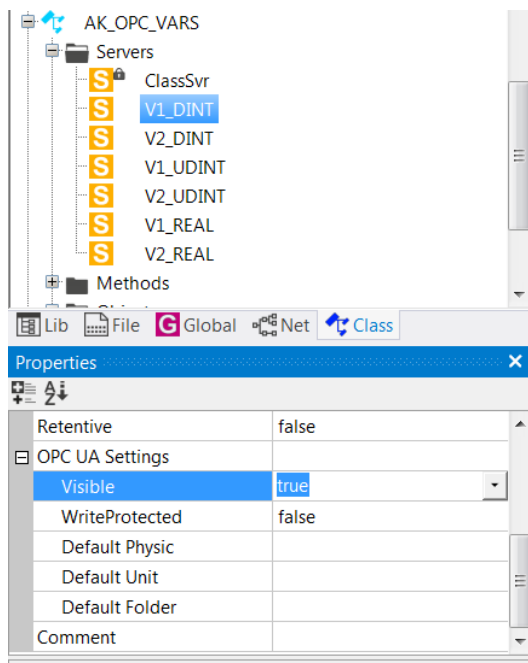
5.2 Implementierung ins SPS-Projekt

Wie bereits erwähnt geschieht der normale Datenaustausch bei SIGMATEK-Systemen über Server-Variablen (Typen DINT, UDINT, REAL und STRING).

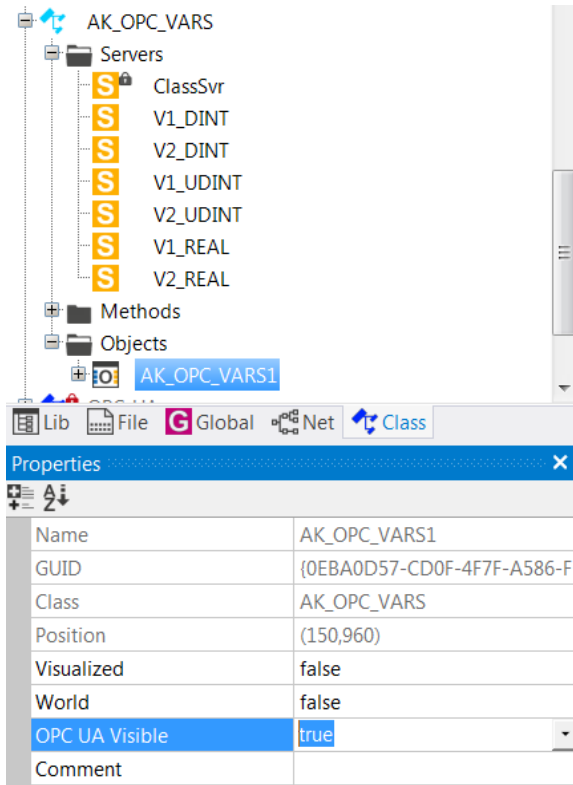
Das SIGMATEK-System baut das OPC_UA.XML-File für den OPC-Server auf.

Dazu müssen folgende Bedingungen erfüllt sein:

- Bei allen gewünschten **Servern** in den Properties das OPC-Attribut "Visible" auf "true" setzen
- Das OPC-Attribut "WriteProtected" je nach Anwendung einstellen



- Bei allen instanziierten **Objekten**, welche Server für die OPC-UA-Datenübertragung beinhalten, das Attribut "**OPC UA Visible**" auf "**true**" setzen



- Übertragen des Projekts in die SPS

Das XML-File wird beim Kompilieren des Programms angelegt und nur beim Download auf die Steuerung übertragen. Bei einem Reset/Start der CPU wird das File weder erstellt noch verändert.

Nun sollte ein OPC-UA Client auf die Variablen zugreifen können.

Zum Testen kann das Programm UaExpert verwendet werden (siehe auch entsprechender Abschnitt in dieser Dokumentation).

6 Konfiguration mittels XML-Datei

Es ist/sind eine oder mehrere anwendungsspezifische Konfigurationsdateien (XML-Format) notwendig, um dem OPC-UA-Server die entsprechenden Datapoints (Variablen) samt Attributen und Verzeichnissen bekanntzugeben.

Wie bereits erwähnt, baut das LASAL Class 2 das OPC-UA.XML-File selbständig auf. Dieses XML-File wird beim Download des Projekts in die CPU übertragen. Aktuell (LASAL V.02.02.153) wird das File in die Root gespeichert. Sollte ein Ordner "C:\OPC-UA" vorhanden sein, so muss das File manuell dorthin kopiert werden.

Um erweiterte Funktionalitäten zu ermöglichen, kann aber auch ein manuell erstelltes XML-File hilfreich sein. In diesem Falle muss jedoch die Projekt-Einstellung "Enable OPC UA" abgeschaltet werden!

Eine XML-Datei muss auf der entsprechenden SPS, auf welcher der OPC-UA-Server läuft, abgelegt sein. Ein OPC-UA-Client kann nur auf derart konfigurierte Datenpunkte zugreifen!

Die Datei ist per Standard mit "OPC-UA.xml" benannt, welche sich im Ordner "C:\OPC-UA" befinden muss. Fehlt dieser Ordner, so muss das File in der Root des Laufwerkes C liegen. Mittels Überladen der virtuellen Methode OPC-UA::FunctSetUp können Name und Pfad geändert werden.

6.1 Grundsätzlicher Aufbau

OPC-UA.XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Config Version="1.0">
  <Trace TraceLevel="40" />
  <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>
```

Alle Konfigurationen werden im Tag "Config" getätigt. Hierbei werden die Sub-Tags "Trace", "Release", "Server" und "DataSet" unterschieden.

Trace	<p>Element ist optional. Es definiert den Level für Trace-Ausgaben. Für mögliche Werte siehe Klassenbeschreibung OPC-UA, Bereich Server - TraceLevel.</p> <p>Tag "Trace" Attribut ...</p> <p><i>TraceLevel</i> legt den Trace Level für Trace-Ausgaben fest</p>
Release	<p>gibt ein oder mehrere freigegebene Verzeichnisse zum File Up-/Download an.</p> <p>Tag "ReleasePath" Attribut ...</p> <p><i>Path</i> legt den/die möglichen Pfade für File-Transfers fest</p>
Server	<p>Element ist optional. Es gibt einen oder mehrere "fremde" OPC-UA-Server an, auf welche die "eigene" OPC-UA-Klasse als Client zugreifen kann. Siehe dazu Kapitel "Client-Datentransfer ..."</p>
DataSet	<p>definiert die einzelnen Datapoints in der SPS mit entsprechenden Attributen. Auf diese Datenpunkte können OPC-UA-Clients zugreifen!</p> <p>Tag "DataElement" mit Attributen ...</p> <p><i>Type</i> (DINT, UDINT, REAL, STRING)</p> <p><i>Hostname</i> umschreibender Name für Dataelement</p> <p><i>Writeprotected</i> (true, false)</p> <p><i>Physic</i> userspezifischer Text - optional</p> <p><i>Unit</i> userspezifischer Text - optional</p> <p><i>Folder</i> userspezifischer Folder - optional</p> <p><i>Label</i> eigentlicher Name des Dataelements (Objekt.Server)</p> <p><i>IsAvailable</i> optional - legt fest, ob der Node im Adressraum sichtbar sein soll <i>IsAvailable</i>="False" ... fix unsichtbar <i>IsAvailable</i>="True" ... fix sichtbar <i>IsAvailable</i>="Object.Server" ... via Klassenserver (0, 1)</p> <p><i>Userrole</i> optional - gibt maximal 32 Schreib-Berechtigungen an (Bit 0 bis Bit 31), siehe auch Userverwaltung OPC-UA-AddOn) <i>Userrole</i>="0" ... kein Schreibrecht gesetzt <i>Userrole</i>="-1" ... alle Schreibrechte gesetzt <i>Userrole</i>="x" ... x = Wert für beliebige Bitmaske</p> <p>Es gibt auch noch zusätzliche Attribute zur Definition der Nodes in den "fremden" OPC-UA-Servern im Falle des optionalen Client-Datentransfers. Siehe dazu Kapitel "Client-Datentransfer ..."</p>

6.2 Beispiel für OPC-UA-Variablen mit 2 Konfigurationsdateien

Config1.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 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="MQ_00" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1\Level2\Level3" Label="Machine0.Server0" />
    <DataElement Hostname="MQ_01" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1\Level2\Level3" Label="Machine0.Server1" />
    <DataElement Hostname="MQ_02" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1\Level2" Label="Machine0.Server2" />
    <DataElement Hostname="MQ_03" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1\Level2" Label="Machine0.Server3" />
    <DataElement Hostname="MQ_04" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1" Label="Machine0.Server4" />
    <DataElement Hostname="MQ_05" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1" Label="Machine0.Server5" />
    <DataElement Hostname="MQ_06" Type="DINT" Writeprotected="false" Physic="" Unit="SEC" Folder="Level1" Label="Machine0.Server6" />
  </DataSet>
</Config>
```

Hinweis zu Strings

Labels für Strings müssen immer mit dem "Data"-Server des String-Objektes angegeben werden.

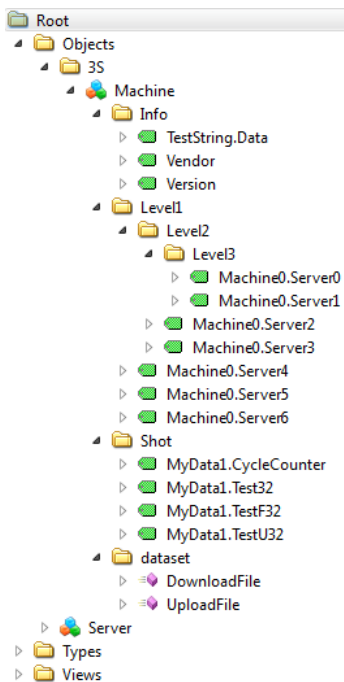
Label="StrSerialNumber.Data"

Dies ist insbesondere bei verschachtelten Objekten zu berücksichtigen, bei welchen der Data-Server "hinausgezogen" wurde.

Label="Testvars.StrSerialNumber.Data"

Adressraum des Servers

Die beiden oben gezeigten Konfigurationsdateien bewirken folgenden Adressraum:



6.3 Übertragung komplexer Daten: ByteString

Dieser Absatz behandelt die Übertragung von ByteString konstanter und variabler Länge.

6.3.1 ByteStrings konstanter Länge

Ab Version 4.0 ist auch die Übertragung komplexer Datentypen möglich. Hier wird der ByteString erläutert. Die nachfolgenden Punkte beziehen sich auf einen ByteString fixer Länge. Die Unterschiede zu einem ByteString variabler Länge werden weiter unten dargestellt.

- Ein ByteString kann verwendet werden, um binäre Daten zu übertragen, die in der Applikation interpretiert werden müssen.
- Die dazu notwendige Variable muss innerhalb einer Anwender-Klasse als Member-Variable definiert werden.
- Die Adresse der Member-Variable muss per Anwenderprogramm auf einen Klassen-Server geschrieben werden.
Dieser Klassen-Server ist in der XML-Datei als Attribut **Label** bekannt zu geben.
- Der **Type** ist hier "BYTESTRING"
- In der XML-Datei muss zusätzlich zu den bereits erläuterten Attributen ein Attribut namens **Size** eingetragen werden. Dieses legt die Länge des ByteStrings fest und kann sich zur Laufzeit nicht mehr verändern, unabhängig von den übertragenen Daten. Dies kann dazu führen, dass Daten entweder abgeschnitten werden oder bei unbekannter Länge die Daten nicht konsistent sind. Siehe hierzu ByteStrings mit variabler Länge, weiter unten.
- Ein weiteres Attribut namens **MemoryAccessMode** wird benötigt. Dieses legt den Zugriff fest und ist hier fix auf "1" zu setzen.
 - Server (0): Wenn der Modus auf "MemoryAccessMode_Server" eingestellt ist, wird der empfangene Wert direkt auf den im entsprechenden Mapping angegebenen Server geschrieben.
 - ServerToMemory (1): Wenn der Modus "MemoryAccessMode_ServerToMemory" eingestellt ist, hält der Server in der Klasse die Adresse eines Speicherblocks. Der Benutzer muss dafür sorgen, dass genügend Speicherplatz zugewiesen wird, um die angegebenen Daten an diese Adresse im Speicher zu schreiben.
 - Memory (2): Der Modus "MemoryAccessMode_Memory" kann nicht in einem Mapping verwendet werden. Bei einem Aufruf einer z. B. Get-Funktion soll der Parameter lasal id direkt der Speicherblock ohne Offset sein. In diesem Fall werden keine Prüfungen mehr auf die Adresse durchgeführt.

Beispiel für einen XML-Eintrag:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config Version="1.1">
  <Release>
    <ReleasePath Path="C:\OPCUA\"/>
  </Release>
  <DataSet>
    <DataElement Hostname="TestByteString" Type="BYTESTRING" Mode="RW" Unit=""
      Label="TestBytestring.Addr" Size="100" MemoryAccessMode="1"/>
  </DataSet>
</Config>
```

6.3.2 ByteStrings variabler Länge

Für einen ByteString variabler Länge muss ein Objekt der Klasse OPC-UA_ByteString platziert werden. Diese wird mit dem Objekt der Klasse OPC-UA verbunden. Das Attribut **Label** zeigt in diesem Fall nicht auf einen Server, sondern auf den Namen des Objekts direkt. Zudem wird keine Länge (**Size**) angegeben. Der MemoryAccessMode kann für diesen Fall weggelassen werden oder wird mit 0 angegeben.

Auf die Daten des ByteStrings kann über die Methoden von MerkerEx zugegriffen werden. Weitere Erläuterungen sind in der entsprechenden Klassendokumentation zu finden.

Beispiel für einen XML-Eintrag:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config Version="1.1">
  <Release>
    <ReleasePath Path="C:\OPCUA\"/>
  </Release>
  <DataSet>
    <DataElement Hostname="TestByteStringDyn" Type="BYTESTRING" Mode="RW" Unit=""
      Label="TestBytestringDyn"/>
  </DataSet>
</Config>
```

6.4 Übertragung komplexer Daten: Strukturierte Variable

Ab Version 4.0 ist auch die Übertragung komplexer Datentypen möglich, hier wird eine strukturierte Variable erläutert.

- Um strukturierte Variablen übertragen zu können, muss der Aufbau der Struktur auch im OPC_UA-Adressraum bekannt sein!
Der Aufbau der strukturierten Variable wird über eine **Model-Datei** beschrieben (z.B. mit UA-Modeler erstellt).

Neben einer **Model.xml** benötigt man auch noch eine auch eine **Mapping.xml**, welche mittels eines **OPC_UA_AddressSpace-Modules** einzulesen sind.

Nähere Informationen und ein Beispiel dazu befinden sich in der Client-Dokumentation und im Client-Demo-Projekt.

- Auch Arrays und Struktur-Arrays sind möglich
- Der Aufbau der Variable muss Server- und Client-seitig absolut identisch sein
- Pointer auf Daten und Texte innerhalb der Variable werden nicht unterstützt. Texte müssen also in der Variable als ARRAY OF CHAR angelegt werden.
- Die dazu notwendige Variable muss innerhalb einer Anwender-Klasse als Member-Variable definiert werden.
- Die Adresse der Member-Variable muss per Anwenderprogramm auf einen Klassen-Server geschrieben werden.
Dieser Klassen-Server ist im Mapping-File als **Label** bekannt zu geben.

Beispiel für ein XML-Eintrag im Mapping-File:

```
<?xml version="1.0" encoding="utf-8"?>
<DataMapping>
  <NamespaceUri>
    <Uri>http://www.sigmatek-automation.com/CA/ </Uri>
  </NamespaceUri>
  <Mappings>
    <Mapping NodeId="ns1;i=6023" Label="OPC_UA_AddressSpace_Demo1.DataPtr" StructureTypeId="3002" Count="1" Size="26518" DataTypeId="22" MemoryAccessMode="1" />
  </Mappings>
</DataMapping>
```

- Unter **<NamespaceUri>** muss der Namespace bekannt gegeben werden.
Dieser muss übereinstimmen mit jenem Namespace, welcher im Zuge der XML-Erstellung für die Adressraum-Instanzen vergeben wurde (siehe Model.xml).
- Die Mapping-Einträge müssen im Element **<Mappings>** erfolgen.
Dabei ist für jede notwendige Verknüpfung eine separate Zeile zu erstellen.
Das Format ist strikt einzuhalten.

- Die **NodeId** ist jene der entsprechenden Adressraum-Instanz
- Der **Label** gibt den Namen der Variable auf der eigenen Station an
Dies ist der Name jenes Klassen-Servers, auf welchen die Adresse der Member-Variable zu finden ist (diese Adresse muss anwenderseitig in der Applikation auf den Server geschrieben werden).
- Der **MemoryAccessMode** legt den Zugriff fest – hier fix "1"
 - Server (0): Wenn der Modus auf "MemoryAccessMode_Server" eingestellt ist, wird der empfangene Wert direkt auf den im entsprechenden Mapping angegebenen Server geschrieben.
 - ServerToMemory (1): Wenn der Modus "MemoryAccessMode_ServerToMemory" eingestellt ist, hält der Server in der Klasse die Adresse eines Speicherblocks. Der Benutzer muss dafür sorgen, dass genügend Speicherplatz zugewiesen wird, um die angegebenen Daten an diese Adresse im Speicher zu schreiben.
 - Memory (2): Der Modus "MemoryAccessMode_Memory" kann nicht in einem Mapping verwendet werden. Bei einem Aufruf einer z. B. Get-Funktion soll der Parameter lasal id direkt der Speicherblock ohne Offset sein. In diesem Fall werden keine Prüfungen mehr auf die Adresse durchgeführt.
- Die **StructureTypeId** gibt die numerische ID des Datentyps in der Model.xml an (kann frei gewählt werden)
- Der **Count** gibt an, wie viele Elemente im Array vorhanden sind (1 bis x)
- Die **Size** gibt an, wie viele Bytes ein einzelnes Array-Element hat (1 bis x)
- Die **DataTypeId** bestimmt den OPC-UA-seitigen Datentyp
Diese IDs sind seitens OPC-UA festgelegt – hier fix "22" für Struct

7 OPC-UA Funktionen

7.1 Lesen/Schreiben von Variablen

Innerhalb dieser Dokumentation wurden das Lesen und Schreiben von Variablen bereits behandelt.

Es können alle Server eines LASAL-Projekts per Konfiguration am OPC-UA-Server für OPC-UA-Clients zur Verfügung gestellt werden. Über die Konfiguration kann eingestellt werden, welche Variablen nur gelesen bzw. auch geschrieben werden können.

Derzeit werden die Datentypen DINT, UDINT, REAL und STRING unterstützt.

Empfehlung:

Da der Zugriff auf Dataentries vom OPC-UA-Server nicht anwenderspezifisch getriggert werden kann, empfiehlt sich ein entsprechendes Interface zwischen OPC-UA-Server und SPS.

7.2 MonitoredItems

Für jede OPC-UA Variable, welche vom Client gelesen werden darf, kann clientseitig auch ein MonitoredItem angelegt werden.

Das eigentliche Monitoring solcher Items geschieht jedoch Server-seitig.

Die Refresh-Zeit für Subscription/Monitored Items wird zwischen Client und Server ausgehandelt. Unser Server unterstützt hierbei minimal 50 ms.

Pro angelegter Subscription ist es möglich, 1000 MonitoredItems hinzuzufügen. Müssen mehr als 1000 Elemente gemonitored werden, so müssen diese auf mehrere Subscriptions aufgeteilt werden.

Empfehlung:

Da jedes MonitoredItem vom Server intern auf Änderung überwacht werden muss, belastet dies das Gesamtsystem.

Daher sollten nur so viele MonitoredItems wie unbedingt notwendig angelegt werden. In der Regel gibt es im Gesamtsystem eine Triggervariable. Anhand dieser Variable kann erkannt werden, ob sich Daten geändert haben oder nicht (z.B. Zykluszähler).

Für dieses Feld soll ein MonitoredItem angelegt werden, damit der Client vom Server automatisch per Event über Änderungen informiert wird. Nach einer Änderung des Triggers kann der Client die geänderten Daten mit einem einzigen Lesevorgang lesen.

7.3 Datei Download

Mit dieser Standard OPC-UA Methode kann eine einzelne Datei vom OPC-UA-Client an den OPC-UA-Server übertragen werden.

Seitens der Server-Anwendung ist Hintergrundwissen nicht nötig.

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

Filename	Name, unter dem die Datei auf der SPS abgelegt wird. Dem Namen kann optional auch ein absoluter Pfad vorangestellt werden. Wird kein Pfad angegeben, so wird dem Filenamen der Pfad "C:\OPCUA\" vorangestellt. Bei Angabe eines absoluten Pfades wird dieser anhand der in der Konfiguration freigegebenen Pfade validiert. Es sind nur Pfade aus der Konfiguration und "C:\OPCUA" gültig.
Data	Enthält den Inhalt der Datei in Form eines ByteStrings.
StatusCode	Rückgabewert informiert über den Erfolg / Misserfolg des Methodenaufrufs.

7.4 Datei Upload

Mit dieser Standard OPC-UA Methode kann eine einzelne Datei vom OPC-UA-Server gelesen und an den OPC-UA-Client übertragen werden.

Seitens der Server-Anwendung ist Hintergrundwissen nicht nötig.

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

Filename	Name, unter dem die Datei auf der SPS gesucht wird. Dem Namen kann optional auch ein absoluter Pfad vorangestellt werden. Wird kein Pfad angegeben, so wird dem Filenamen der Pfad "C:\OPCUA\" vorangestellt. Bei Angabe eines absoluten Pfades wird dieser anhand der in der Konfiguration freigegebenen Pfade validiert. Es sind nur Pfade aus der Konfiguration und "C:\OPCUA" gültig.
Data	Enthält den Inhalt der Datei in Form eines ByteStrings.
StatusCode	Rückgabewert informiert über den Erfolg / Misserfolg des Methodenaufrufs.

8 Klasse OPC-UA_Server



Für die Funktionalität des OPC-UA-Servers muss eine Instanz dieser Klasse in einem Netzwerk platziert werden.

Beim Programmstart wird ein eigener Thread angelegt, in welchem der OPC-UA-Server läuft. Es sind hierfür keine weiteren programmtechnischen Tätigkeiten zu erledigen.

8.1 Schnittstellen

8.1.1 Server

ClassSrv	ClassSrv
AuthentMode	Authentifizierungs-Mode 0 ... Authentifizierung via Username/Password (mittels File UserConfiguration.xml) 1 ... Authentifizierung ist auf "anonym" geschaltet
CycTimeAct	Zykluszeit aktuell [µs]
CycTimeMin	Zykluszeit Minimum [µs]
CycTimeMax	Zykluszeit Maximum [µs]
CycTimeAvg	Zykluszeit Mittelwert [µs]
ResetCycTimes	Befehl "Reset Cycle Times" Das Setzen auf #1 führt zum Reset der oben genannten Zykluszeiten

8.1.2 Clients

OPC-UA	Objekt Kanal zum OPC-UA-Objekt.
config	<p>Bitmuster für Konfiguration</p> <p>Bit 0 .. Enable historical data 0 = "historical data ist disabled" / 1 = "historical data ist enabled"</p> <p>Bit 1 .. Enable historical events 0 = "historical events disabled" / 1 = "historical events enabled"</p> <p>Bit 2 .. Reserve</p> <p>Bit 3 .. Reserve</p> <p>Bit 4 .. Verwende alte Namespace URI 0 = "use the current URI" / 1 = "use the old URI"</p> <p>Hinweis: Die alte URI darf nur aus Kompatibilitätsgründen verwendet werden. In jedem anderen Fall soll dieses Bit auf 0 gesetzt werden!</p>
EncryptionConfig	<p>Bitmuster für Konfiguration der Verschlüsselungs-Unterstützung 0 = "Disabled" / 1 = "Enabled"</p> <p>Bit 0 ... SecurityPolicy None/SecurityMode None Bit 1 ... SecurityPolicy Basic256/SecurityMode Sign Bit 2 ... SecurityPolicy Basic256/SecurityMode Sign and Encrypt Bit 3 ... SecurityPolicy Basic256Sha256/SecurityMode Sign Bit 4 ... SecurityPolicy Basic256Sha256/SecurityMode Sign and Encrypt Bit 5 ... SecurityPolicy Aes128Sha256Rsa0aep/SecurityMode Sign Bit 6 ... SecurityPolicy Aes128Sha256Rsa0aep/SecurityMode Sign and Encrypt Bit 7 ... Reserved Bit 8 ... Reserved</p> <p>Hinweis: es dürfen nur jene Verschlüsselungen aktiviert werden, welche auch tatsächlich vom verwendeten OS unterstützt werden! RTK unterstützt kein SSL! SSL wird grundsätzlich ab Salamander Version 09.03.116. Basic256Sha256 und Aes128Sha256Rsa0aep werden ab Salamander Version 09.03.160 unterstützt.</p>

8.1.3 Globale Methoden

Init	Initialisierungen und Anlegen des OPC-UA-Threads
OpcUaThread	OPC-UA Dienste werden in diesem Thread abgearbeitet.
FunctStart	Wird beim Starten des OPC-UA-Servers einmalig aufgerufen und signalisiert dem Anwender, dass dieser Dienst erfolgreich gestartet wurde.
FunctRun	Wird zyklisch aufgerufen, sofern der Dienst gestartet wurde.
FunctSetUp	<p>Mit dieser Methode werden dem OPC-UA-Server die entsprechenden Konfigurationsdateien (.XML) bekanntgegeben. Sie wird ebenfalls einmalig unmittelbar nach dem Start aufgerufen. Es dürfen auch mehrere verschiedene Konfigurationsdateien bekanntgegeben werden. Dies geschieht über den Aufruf der Funktion</p> <p>OUT retcode 0= OK ansonsten Fehlercode Fehlercode (Server wird in diesem Fall nicht gestartet!)</p> <p>Die Basisimplementierung lädt hier die Standardkonfigurationsdatei "OPC-UA.XML". Somit entspricht retcode dem Rückgabewert von AddXmlConfig.</p>
RegisterProvider	<p>Registriert die Provider von allen Modulen</p> <p>OUT retcode 0 ... OK / <> 0 ... RegisterProvider() fehlerhaft</p>
ProviderRun	Server-Provider-Funktion - wird vom OPC-UA-Dienst einmal pro Zyklus aufgerufen.

<p>SetParameter</p>	<p>Kann verwendet werden, um ablaufspezifische Parameter zu setzen.</p> <p>IN ParaNr Parameternummer IN Value neuer Wert für den gewählten Parameter OUT retcode 0 = Parameter erfolgreich gesetzt -1 = Parameter wurde nicht gesetzt (falscher Wert, ...)</p> <p>Gültige Parameter:</p> <p>0 = OPC_SET_OPTION ... setzen einer Option Value: Pointer auf den Option-String</p> <p>Option-Strings zum Ausblenden von Funktionen aus OPC_UA-Adressraum:</p> <p>"DISABLE:ACTIVATE_DATASET_EXT_INFO" "DISABLE:PREPARE_DATASET_EXT_INFO" "DISABLE:DOWNLOAD_FILE_EXT_INFO" "DISABLE:UPLOAD_FILE_EXT_INFO" "DISABLE:GET_ALL_ACTIVE_STANDARD_ALARMS"</p> <p>1 = OPC_UA_PAR_SET_PORTNR ... Port Nummer für die IP-Adresse Value: 1 = 1024 bis 49151 Achtung: muss hierfür im Init VOR dem _FirstScan aufgerufen werden!</p> <p>2 = OPC_UA_PAR_SET_SENDCERT Der Dienst sendet kein Serverzertifikat für nicht sichere Verbindungen</p> <p>3 = OPC_UA_PAR_SET_DATA_LOGGER_MAX_ITEMS Legt die maximale Anzahl an Elementen für das Data Logging fest</p>
<p>GetNamespaceIndexByUri</p>	<p>Liefert den Namespace-Index für die gewünschten Namespace-URI.</p> <p>IN namespaceUri Namespace-URI-String des gewünschten Namespace OUT namespaceIndex Namespace Index des gewünschten Namespace -1 = Der gewünschte Namespace ist nicht im Server-Namespace-Array registriert</p>
<p>SetDisplayName</p>	<p>Setzt den DisplayName des Zielknotens.</p> <p>IN nodeId ID des Zielknotens IN locale Locale des Anzeigenamens IN displayName Übersetzter Anzeigename (entsprechend dem Locale) OUT result 0 = kein Error / <> 0 = OpcUa Error-Code</p>

SetBrowseName	<p>Setzt den BrowseName des Zielknotens.</p> <p>IN nodeId ID des Zielknotens</p> <p>IN namespaceIndex Namespace-Index des BrowseName</p> <p>IN browseName BrowseName String</p> <p>OUT result 0 = kein Error / <> 0 = OpcUa Error-Code</p>
GetCycTimeMin	retourniert den aktuellen Wert des Servers "CycTimeMin"
GetCycTimeMax	retourniert den aktuellen Wert des Servers "CycTimeMax"
GetCycTimeAvg	retourniert den aktuellen Wert des Servers "CycTimeAvg"
ResetCycleTimes	Der Aufruf dieser Methode führt zum Reset aller Zykluszeiten.
GetCurrentSessionCount	OPC_UA Diagnose - returns CurrentSessionCount
GetCumulatedSessionCount	GetCumulatedSessionCount
GetReadCount	OPC_UA Diagnose - retourniert die Summe aller ReadCount aller aktuell geöffneten Sitzungen aller Clients
GetWriteCount	OPC_UA Diagnose - retourniert die Summe aller WriteCount aller aktuell geöffneten Sitzungen aller Clients
GetBytesRead	OPC_UA Diagnose - retourniert die Summe aller Read Bytes aller aktuell geöffneten Sitzungen aller Clients
GetBytesWritten	OPC_UA Diagnose - retourniert die Summe aller Written Bytes aller aktuell geöffneten Sitzungen aller Clients
GetTotalNodesCount	OPC_UA Diagnose - retourniert TotalNodesCount
GetRejectedSessionCount	OPC_UA Diagnose - retourniert RejectedSessionCount
GetRejectedRequestsCount	OPC_UA Diagnose - retourniert RejectedRequestsCount
GetSessionAbortCount	OPC_UA Diagnose - retourniert SessionAbortCount
GetSessionTimeoutCount	OPC_UA Diagnose - retourniert SessionTimeoutCount
GetCumulatedAccessCount	OPC_UA Diagnose - retourniert "CumulatedAccessCount" (Summe aller Arten von Client-Zugriffen, wie read, write, call, ...)
AlarmChanged StandardAlarmChanged	<p>Mit dieser Methode kann eine Änderung an einem Alarm an den OPC-UA Server übergeben werden. Jede Änderung an einem Alarm muss gemeldet werden. Sowohl die Aktivierung als auch die Deaktivierung eines Alarmes. Über diese Methodik wird die Liste der aktuellen Alarme aktuell gehalten. Dieser Aufruf muss zusammen mit allen Alarm-bezogenen Funktionen threadsicher durchgeführt werden.</p> <p>IN alarm Informationen über den geänderten Alarm</p> <p>OUT state 0</p>

AlarmChangedUC	<p>Entspricht der Methode "AlarmChanged". Der Unterschied liegt darin, dass Strings bei dieser Methode als Array von 16-bit Werten übergeben werden. Somit können beliebige Unicode Zeichen übertragen werden.</p> <p>Dieser Aufruf muss zusammen mit allen Alarm-bezogenen Funktionen threadsicher durchgeführt werden.</p> <p>IN alarm Informationen über den geänderten Alarm</p> <p>OUT retcode 0</p>
DatasetPreparationFinished	<p>Mit dieser Methode kann dem OPC-UA Server mitgeteilt werden, dass die Aufbereitung eines Einstell Datensatzes für die Übertragung an einen Client abgeschlossen wurde.</p> <p>IN status Status der Aktivierung (0: Fehlerfrei, !=0: Fehlercode)</p> <p>IN datasetId Eindeutige Id der Übertragung / des Datensatzes</p> <p>IN datasetName Name des Einstell Datensatzes</p> <p>IN path Pfad in dem die zugehörige Datei gespeichert wurde</p> <p>OUT state 0</p> <p>OPC-UA Clients können sich mittels OPC-UA Event über die Aufbereitung von Einstell Datensätzen benachrichtigen lassen. Im Anschluss daran, kann ein Client den Einstell Datensatz über die Methode "DownloadFile" vom OPC-UA Server lesen.</p>
DatasetActivationFinished	<p>Mit dieser Methode kann dem OPC-UA Server die Aktivierung eines Einstell Datensatzes bekannt gegeben werden.</p> <p>IN status Status der Aktivierung (0: Fehlerfrei, !=0: Fehlercode)</p> <p>IN datasetId eindeutige Id der Übertragung / des Datensatzes</p> <p>IN datasetName Name des Einstell Datensatzes</p> <p>IN path Pfad in dem die zugehörige Datei gespeichert wurde</p> <p>OUT state 0</p> <p>OPC-UA Clients können sich mittels OPC-UA Event über die Aktivierung von Einstell Datensätzen benachrichtigen lassen.</p>
AllActiveAlarms	<p>Mit dieser Methode kann die Liste der aktiven Alarmer an den OPC-UA Server übergeben werden. Diese Methode muss nur einmalig, bei Programmstart, aufgerufen werden, damit die Liste der aktiven Alarmer initialisiert werden kann.</p> <p>Dieser Aufruf muss zusammen mit allen Alarm bezogenen Funktionen threadsicher durchgeführt werden.</p> <p>IN alarmList Zeiger auf die Liste der aktiven Alarmer</p> <p>IN listCount Anzahl der Alarmer in der Liste</p> <p>OUT state 0</p> <p>OPC-UA Clients können über die Funktion "GetAllActiveAlarms" die Liste der aktiven Alarmer abfragen.</p>
RefreshUserConfiguration	<p>Der Aufruf führt zum erneuten Laden der Users-XML Datei.</p>

GetClientDiagnosticInfos	<p>OPC_UA Diagnosis - retourniert Client/Session spezifische Informationen mit Hilfe eines Pointers auf einen strukturierten Speicher.</p> <p>ACHTUNG: nach dem Aufruf dieser Methode muss nach dem Auswerten der Daten zwingend die Methode "ClearClientDiagnosticInfos()" aufgerufen werden, damit der Speicher wieder freigegeben wird.</p>																				
ClearClientDiagnosticInfos	Gibt den Speicher frei, welcher durch die Methode "GetClientDiagnosticInfos()" allokiert wurde.																				
CreateAddressSpaceDescription	Methode zum Erstellen und Exportieren der "address space information" in ein XML-File namens "model_definition.xml"																				
LogHistoryData	<p>Diese Funktion wird immer dann aufgerufen, wenn ein Datenpunkt für den historischen Zugriff protokolliert werden muss. In dieser Funktion wird die DataLoggerBase-Klasse aufgerufen, wenn eine Verbindung besteht.</p> <table> <tr> <td>IN primaryKey</td><td>Der Primärschlüssel des zu speichernden Nodes</td></tr> <tr> <td>IN statusCode</td><td>Der Statuscode des zu speichernden Werts</td></tr> <tr> <td>IN sourceTime</td><td>Quellzeitstempel des zu speichernden Werts</td></tr> <tr> <td>IN serverTime</td><td>Server-Zeitstempel des zu speichernden Werts</td></tr> <tr> <td>IN valueType</td><td>Der Typ des zu speichernden Werts</td></tr> <tr> <td>IN dataLength</td><td>Die Länge des zu speichernden Bytestrings</td></tr> <tr> <td>IN data</td><td>Der Pointer auf den zu speichernde Bytestring</td></tr> <tr> <td>IN userData</td><td>Zusätzlicher Parameter, der später verwendet werden kann, um den richtigen Logger für den angegebenen Datenwert oder weitere hilfreiche Informationen verwenden zu können.</td></tr> <tr> <td>OUT retcode</td><td>Rückgabewert zum Speichern</td></tr> <tr> <td></td><td>0 = Speichern OK / <> 0 = Fehler mit Fehlernummer</td></tr> </table>	IN primaryKey	Der Primärschlüssel des zu speichernden Nodes	IN statusCode	Der Statuscode des zu speichernden Werts	IN sourceTime	Quellzeitstempel des zu speichernden Werts	IN serverTime	Server-Zeitstempel des zu speichernden Werts	IN valueType	Der Typ des zu speichernden Werts	IN dataLength	Die Länge des zu speichernden Bytestrings	IN data	Der Pointer auf den zu speichernde Bytestring	IN userData	Zusätzlicher Parameter, der später verwendet werden kann, um den richtigen Logger für den angegebenen Datenwert oder weitere hilfreiche Informationen verwenden zu können.	OUT retcode	Rückgabewert zum Speichern		0 = Speichern OK / <> 0 = Fehler mit Fehlernummer
IN primaryKey	Der Primärschlüssel des zu speichernden Nodes																				
IN statusCode	Der Statuscode des zu speichernden Werts																				
IN sourceTime	Quellzeitstempel des zu speichernden Werts																				
IN serverTime	Server-Zeitstempel des zu speichernden Werts																				
IN valueType	Der Typ des zu speichernden Werts																				
IN dataLength	Die Länge des zu speichernden Bytestrings																				
IN data	Der Pointer auf den zu speichernde Bytestring																				
IN userData	Zusätzlicher Parameter, der später verwendet werden kann, um den richtigen Logger für den angegebenen Datenwert oder weitere hilfreiche Informationen verwenden zu können.																				
OUT retcode	Rückgabewert zum Speichern																				
	0 = Speichern OK / <> 0 = Fehler mit Fehlernummer																				

<p>ReadHistoryData</p>	<p>Diese Funktion wird immer dann aufgerufen, wenn ein Verlauf gelesen werden muss.</p> <p>IN primaryKey Primärschlüssel des zu lesenden Nodes</p> <p>IN startTime Startzeit der Werte des zu lesenden Nodes</p> <p>IN endTime Endzeit der Werte des zu lesenden Nodes</p> <p>IN isInverse Dient dem Lesen der Daten in umgekehrter Richtung</p> <p>IN numValues Pointer auf die Anzahl der zu lesenden Werte</p> <p>IN results Pointer darauf, wo die Ergebnisse gespeichert werden sollen, der Speicher ist bereits allokiert</p> <p>IN continuationPoint Zeiger auf den Fortsetzungspunkt, der bei Bedarf gesetzt werden muss Der Fortsetzungspunkt enthält einen Zeitstempel, von dem die letzten Werte geschrieben wurden, und einen Fortsetzungsversatz, der die Anzahl der Werte anzeigt, die genau zur gleichen Zeit gelesen wurden</p> <p>IN continuationOffset Der Fortsetzungsoffset von der Anfrage. Wenn festgelegt, dann wird die Anzahl der Werte indiziert, die genau zur gleichen Startzeit gelesen wurden</p> <p>IN userData Zusätzlicher Parameter, der später verwendet werden kann, um den richtigen Logger zum Lesen oder weitere Informationen zu finden, die hilfreich sein könnten.</p> <p>OUT retcode Rückgabewert des Lesens 0 = Lesen OK / <>0 = Error (z.B. out of memory)</p>
<p>LogHistoryEvent</p>	<p>Diese Funktion wird immer dann aufgerufen, wenn ein Ereignis für den historischen Zugriff protokolliert werden muss. In dieser Funktion wird die DataLoggerBase-Klasse aufgerufen, wenn eine Verbindung besteht.</p> <p>Achtung: Diese Methode ist aktuell nicht implementiert!</p> <p>IN primaryKey Primärschlüssel des zu speichernden Events</p> <p>IN noOfEvents Anzahl der zu speichernden Events</p> <p>IN variants Variant mit den Events</p> <p>IN userData Zusätzlicher Parameter, der später verwendet werden kann, um den richtigen Logger für den angegebenen Datenwert oder weitere hilfreiche Informationen verwenden zu können.</p> <p>OUT retcode Rückgabewert 0 = OK / <> 0 = Fehler mit Fehlernummer</p>
<p>RegisterDataLogger</p>	<p>Registriert einen Datenlogger auf dem Server, auf dem Daten für den historischen Zugriff gespeichert werden. Der Datenlogger muss eine Implementierung der OPC_UA_DataLoggerBase sein.</p> <p>Die Methode wird in der Regel von der konkreten Implementierung des DataLoggers zum Beispiel im Init aufgerufen.</p> <p>IN pDataLogger Der Pointer auf die Implementierung einer OPC_UA_DataLoggerBase.</p> <p>OUT retcode 0 = OK / -1 = Pointer ist fehlerhaft</p>

RegisterEventSource	<p>Mit dieser Methode wird eine nodeld als Ereignisquelle registriert. Dies ist relevant, wenn die Ereignisse gefiltert werden sollen, d. h. der Notifier ist ein anderer als der Server-Knoten.</p> <p>Ohne Registrierung wird das Ereignis vom Server-Knoten als Notifier gesendet. Wenn registriert, ist der nächstgelegene Elternknoten der angegebenen nodeld der Notifier, wenn ein Ereignis eintritt.</p> <p>Wenn die Registrierung fehlschlägt, wird das Ereignis trotzdem vom Server-Knoten gemeldet.</p> <p>IN Nodeld Die Nodeld, die als Ereignisquelle für den nächsten übergeordneten Knoten, der ein Notifier ist, registriert werden soll.</p> <p>OUT retcode Der Rückgabewert zeigt an, ob das Hinzufügen des Knotens als Quelle erfolgreich war (OpcUa_Good = 0) oder nicht. Für eine detaillierte Fehlerbeschreibung siehe die Liste der Fehlercodes am Ende der Dokumentation des OPC-UA_Client.</p>
----------------------------	---

8.1.4 Private Methoden

OPC_UA_Server	Konstruktor. Initialisiert die OPC_UA Schnittstelle.
AddServerFacets	<p>Fügt die angegebenen Strings als Facets zum ServerProfileArray (i=2269) hinzu. Die Methode prüft nicht, ob die angegebenen URIs eindeutig bzw. gültig sind, sondern übernimmt lediglich den Inhalt.</p> <p>IN facets Das Array an Strings mit Facet-URIs, das hinzugefügt werden soll.</p> <p>IN noOfFacets Die Anzahl der String, die in dem Array übergeben und hinzugefügt werden sollen.</p> <p>OUT result Liefert 0 bei Erfolg oder eine OPC_UA-Fehlernummer.</p>
SetOptions	<p>Mit dieser Funktion können Optionen eingestellt werden, mit denen der Programmablauf verändert werden kann.</p> <p>IN options "USE:HOSTNAME-AS-BROWSENAME" "USE:ALPHANUMERIC-IDENTIFIERS" "DISABLE:ACTIVATE_DATASET_EXT_INFO" "DISABLE:PREPARE_DATASET_EXT_INFO" "DISABLE:DOWNLOAD_FILE_EXT_INFO" "DISABLE:UPLOAD_FILE_EXT_INFO" "DISABLE:GET_ALL_ACTIVE_STANDARD_ALARMS"</p> <p>OUT retcode 0</p>
Setvalue32Changed	<p>Mit dieser Methode können Änderungen (vom Typ "DINT") am Einstell Datensatz an den OPC-UA Server übergeben werden. Dieser Aufruf muss zusammen mit allen anderen „Setvalue..Changed“ Funktionen threadsicher durchgeführt werden.</p> <p>IN change Allgemeine Eigenschaften der Parameteränderung</p> <p>IN oldValue Wert vor der Änderung</p> <p>IN newValue Aktueller Wert / Wert nach der Änderung</p> <p>OUT state 0</p> <p>OPC-UA Clients können sich mittels OPC-UA Event über Änderungen am Einstell Datensatz benachrichtigen lassen.</p>
SetvalueU32Changed	Äquivalent zur Methode "Setvalue32Changed" für den Datentyp UDINT
SetvalueF32Changed	Äquivalent zur Methode "Setvalue32Changed" für den Datentyp REAL
SetvalueStringChanged	<p>Äquivalent zur Methode "Setvalue32Changed" für den Datentyp CHAR Dieser Aufruf muss zusammen mit allen anderen „Setvalue..Changed“ Funktionen threadsicher durchgeführt werden.</p> <p>IN change Allgemeine Eigenschaften der Parameteränderung</p> <p>IN oldValue Wert vor der Änderung</p> <p>IN newValue Aktueller Wert / Wert nach der Änderung</p> <p>OUT state 0</p>

SetvalueStringChangedUC	<p>Entspricht der Methode "SetvalueStringChanged". Der Unterschied liegt darin, dass der Übergabeparameter bei dieser Methode als Array von 16-bit Werten übergeben wird. Somit können beliebige UniCode Zeichen übertragen werden. Dieser Aufruf muss zusammen mit allen anderen „Setvalue..Changed“ Funktionen threadsicher durchgeführt werden.</p>
InitDatasetWorkingPath	<p>Mit dieser Methode kann der Standardpfad für Operationen mit dem Einstelldatensatz zur Laufzeit festgelegt werden. Dieser Pfad ist neben den freigegebenen Pfaden aus der Konfiguration für alle Dateioperationen gültig. Ist dieser Pfad gesetzt, wird er als Standardpfad für Dateioperationen ohne Pfadangaben verwendet. Z.B: Wenn der DatasetWorkingPath auf "c:\datensatz\" festgelegt wurde, dann wird bei einem Aufruf von UploadFile mit Parameter "test.txt", die Datei im Verzeichnis "c:\datensatz\text.txt" gespeichert.</p> <p>IN path Angabe des Standardpfades OUT retcode 0</p>
InitFileSystemCallback	<p>Mit dieser Methode kann dem OPC-UA Server eine Callback Funktion bereitgestellt werden.</p> <p>IN f_CB_fileSystem Callback für Änderungen am File System OUT retcode 0</p> <p>Der Callback "f_CB_fileSystem" wird aufgerufen, wenn ein Client durch einen Funktionsaufruf eine Veränderung am Dateisystem verursacht. So verursachen z.B. alle Dateifunktionen (Upload File, Download File, Activate Dataset, Prepare Dataset) Änderungen am Dateisystem und haben somit einen Aufruf dieser Callback Funktion zur Folge.</p>
InitDatasetCallback	<p>Mit dieser Methode können dem OPC-UA Server zwei Callback Funktionen bereitgestellt werden.</p> <p>IN f_CB_activateDS CallBack für Aktivierung von Einstelldatensätzen IN f_CB_prepareDS CallBack für Bereitstellung von Einstelldatensätzen OUT retcode 0</p> <p>Der Callback (CB) "f_CB_activateDS" wird aufgerufen, wenn ein Client einen Datensatz an die Steuerung übertragen und aktivieren will. Dieser Callback dient dem Steuerungsprogramm als Anstoss für das Einlesen und die Aktivierung des gewünschten Einstelldatensatzes.</p> <p>Der Callback "f_CB_prepareDS" wird aufgerufen, wenn ein Client einen Datensatz anfordert. Dieser Callback dient dem Steuerungsprogramm als Anstoss für die Bereitstellung des gewünschten Einstelldatensatzes.</p>
InitAlarmCallback	<p>Mit dieser Methode kann dem OPC-UA Server eine Callback Funktion bereitgestellt werden. Diese Callback Funktion wird vom OPC-UA Server während der Initialisierung aufgerufen. Über diese Methode fordert der OPC-UA Server die Übertragung der Liste aller aktiven Alarmer an.</p> <p>IN f_CB_alarmList Zeiger auf die Callback Funktion OUT retcode 0</p>
IniGetStringArrayCallback	<p>Mit dieser Methode kann dem OPC-UA Server eine Callback Funktion zum Lesen von String-Arrays bereitgestellt werden</p>

**IniSetStringArrayCall
back**

Mit dieser Methode kann dem OPC-UA Server eine Callback Funktion zum Schreiben von String-Arrays bereitgestellt werden

8.2 FAQ zu Leistungsdaten und Speicherbedarf

→ wie viel Speicher wird benötigt?

Der OPC_UA-Server (Klassen OPC_UA und OPC_UA_Server) benötigt für den Aufbau von OPC_UA Standard-Adressraum und -Umgebung etwa:

... 5,42 MB Codespeicher

... 4,96 MB RAM (davon 3,72 MB UserHeap)

Der weitere Speicherbedarf ist abhängig von der Anzahl der Datenpunkte.

→ wie viele Datenpunkte unterstützt die Klasse als OPC_UA-Server?

Technisch unbegrenzt, speicherabhängig

→ wie viele Clients können sich auf den OPC_UA-Server verbinden?

Die Anzahl ist auf 50 beschränkt bzw. abhängig vom verfügbaren Speicher.

→ Speicherbedarf je Server-Datenpunkt für externe Clients?

ca. 1.244 Byte pro Datenpunkt für Einlesen des XML-Files, usw.

→ CPU-Last je Server-Datenpunkt und je Verbindung zu externem Client?

1: Die Startzeit des OPC_UA Servers ist stark abhängig von der Anzahl der OPC_UA-Variablen und der eingesetzten CPU - sie kann zwischen wenigen Sekunden bis zu über einer Minute betragen.

2: Die Dauer eines Verbindungsaufbaues zu einem externen Client ist genau wie unter Punkt 1 zu bewerten.

3: Die CPU-Last für den Datenaustausch nach dem Verbinden eines Clients ist hauptsächlich vom Client abhängig

→ arbeitet die Klasse anonym oder mit Anmeldung?

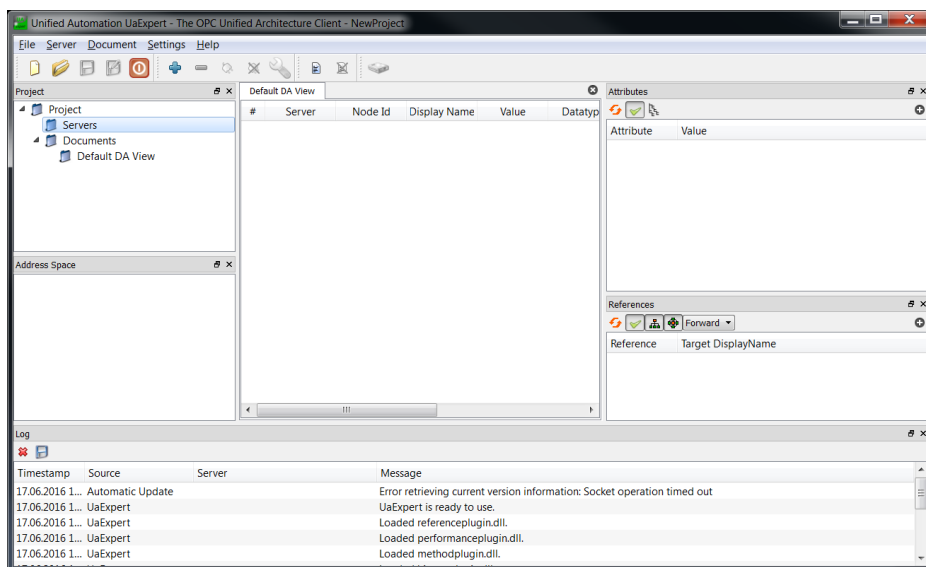
Anonym, mit Benutzername & Passwort und auch mit Verschlüsselung

9 Win-Programm UaExpert

Dieses Programm ist nicht von SIGMATEK und ist für den Echtbetrieb einer OPC-UA-Kommunikation auch nicht notwendig.

Das Tool kann allerdings bei der Erstinbetriebnahme einer OPC-UA-Kommunikation hilfreich sein.

9.1 Verbindung einrichten



Nach dem ersten Start ist kein Projekt vorhanden.

→ Rechtsklick auf "Servers" im Bereich Projekt – Klick auf "Add ..."

weiter auf der nächsten Seite ...

Add Server

Configuration Name: Unified Automation DemoServer

Discovery | Advanced

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

OK Cancel

→ hier muss nur die Endpoint-URL angegeben werden

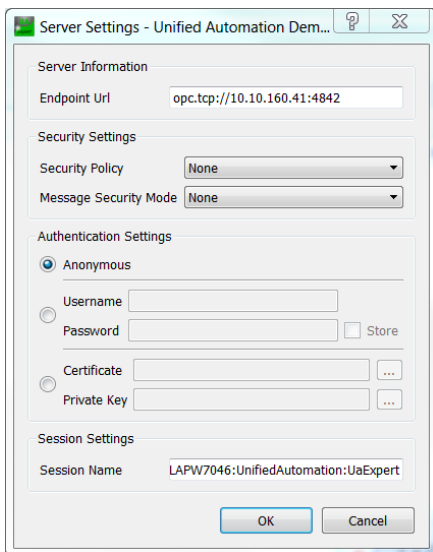
`opc.tcp://10.10.160.41:4842`

- ... die IP-Adresse ist jene der PLC
- ... der Port ist bei Sigmatek fix 4842

9.2 Verbindung aufbauen

Vor dem Verbindungsaufbau sollte in den Server-Settings die Art der Authentifizierung eingestellt werden.

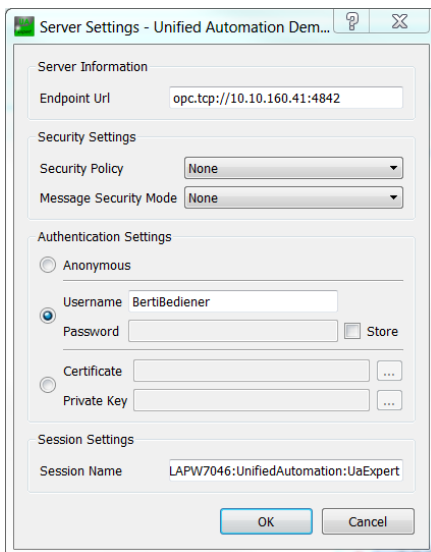
A: anonyme Anmeldung



The screenshot shows the 'Server Settings' dialog box with the following configuration:

- Server Information:** Endpoint Url is `opc.tcp://10.10.160.41:4842`.
- Security Settings:** Security Policy is set to `None` and Message Security Mode is set to `None`.
- Authentication Settings:** The `Anonymous` radio button is selected. The Username, Password, Certificate, and Private Key fields are empty.
- Session Settings:** Session Name is `LAPW7046:UnifiedAutomation:UaExpert`.
- Buttons:** OK and Cancel buttons are at the bottom right.

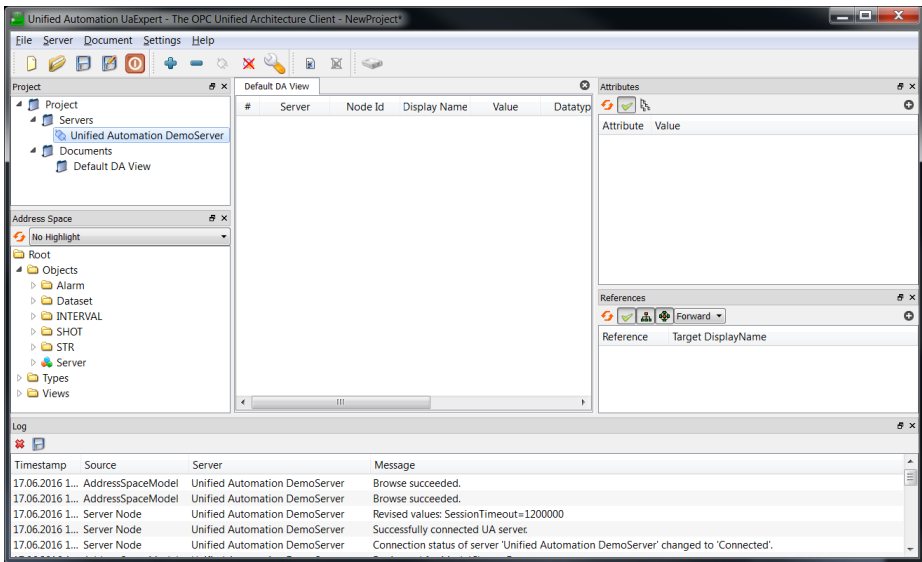
B: Anmeldung mit Benutzer/Passwort



The screenshot shows the 'Server Settings' dialog box with the following configuration:

- Server Information:** Endpoint Url is `opc.tcp://10.10.160.41:4842`.
- Security Settings:** Security Policy is set to `None` and Message Security Mode is set to `None`.
- Authentication Settings:** The `Anonymous` radio button is deselected, and the `Username` radio button is selected. The Username field contains `BertiBediener`. The Password field is empty. The Certificate and Private Key fields are empty.
- Session Settings:** Session Name is `LAPW7046:UnifiedAutomation:UaExpert`.
- Buttons:** OK and Cancel buttons are at the bottom right.

Diese Einstellung muss zu der Art der Authentifizierung des OPC-UA-Servers passen!
Siehe Kapitel "Zugriffsberechtigung" innerhalb dieser Dokumentation.



Mittels eines Rechtsklicks auf den Server und anschließendem Klicken auf "Connect" wird die Verbindung aufgebaut.

Sollte die Anmeldung mit Benutzer/Passwort gewählt sein, so wird man vor dem eigentlichen Connect noch zur Eingabe des Passwortes aufgefordert.

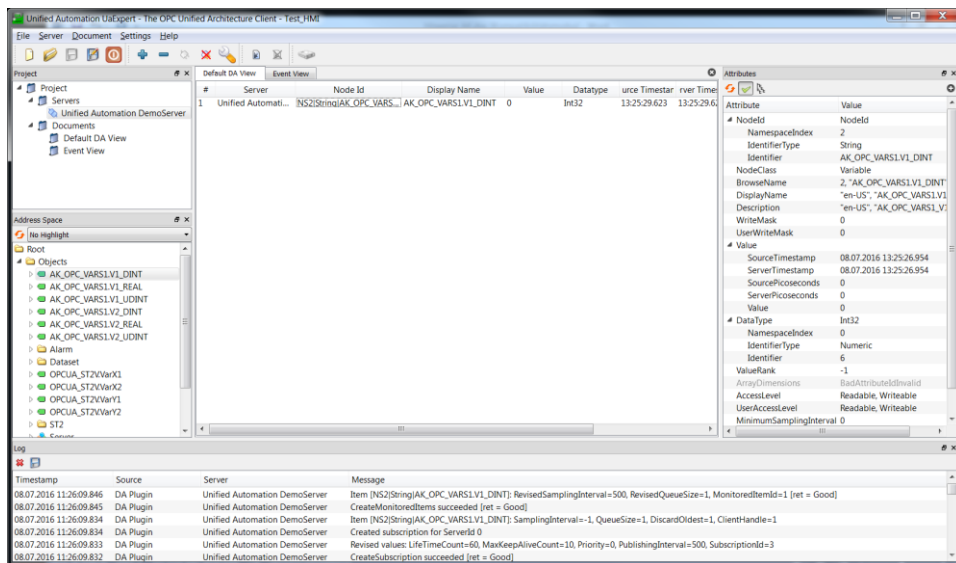
Nun wird vollautomatisch vom OPC-UA-Client die Variablen-Liste vom OPC-Server angefordert und hier zur Anzeige gebracht.

Der OPC-Server baut diese Variablenliste anhand seines eigenen OPC XML-Files auf.

9.3 Datenaustausch

Der normale Datenaustausch geschieht bei SIGMATEK-Systemen über Server-Variablen.

Unterstützt werden momentan: DINT, UDINT, REAL und STRING



- im Bereich "Address Space" auf der linken Seite findet man die Variablen-Liste, welche aus der SPS gelesen wurde
- den Wert der jeweiligen Variablen findet man rechts unter "Value"
- eine Aktualisierung findet hier nur beim Selektieren statt.
- durch "Drag and Drop" kann eine Variable auch in das mittige View-Feld gezogen werden.
- Variablen, welche hier platziert sind, werden zyklisch aktualisiert.
- die Values können geändert werden

9.4 Alarme

Für Alarme bietet OPC-UA ein flexibles Alarmwesen.

Die Basis-Klasse kann allerdings auf die SIGMATEK-Alarme nicht zugreifen.

SIGMATEK bietet dazu eine erweiterte OPC-UA-Klasse an.

9.5 Events

Für Ereignisse bietet OPC-UA ein flexibles Eventwesen.

Die Basis-Klasse kann allerdings auf die SIGMATEK-Ereignisse nicht zugreifen.

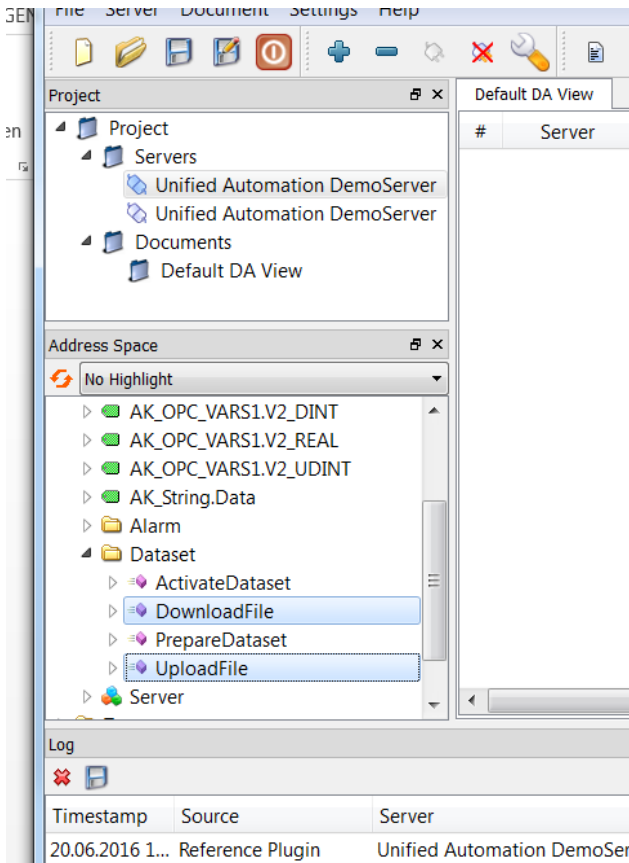
Eine Erweiterung, um auch diese Funktionalität zu unterstützen, ist bereits in Planung.

9.6 File-Transfer

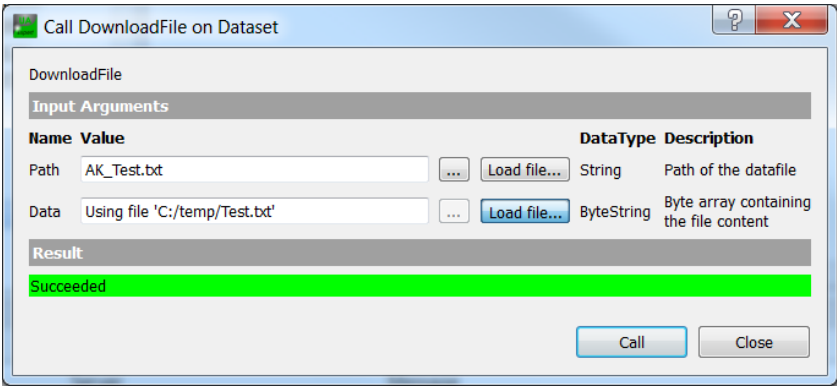
Ein File-Transfer kann in beide Richtungen durchgeführt werden. Er wird jedoch in beiden Fällen vom Client getrieben.

In der Basis-Klasse hat der Server keinerlei Einfluss auf den Zeitpunkt der Übertragung.

Im Programm UaExpert findet man diese Funktionen auf der linken Seite im Bereich "Address Space" im Eintrag "Dataset". Die Funktionen werden mittels Rechtsklick darauf und der Auswahl "Call" aufgerufen.



9.6.1 File von Client (Win) auf Server (SPS)

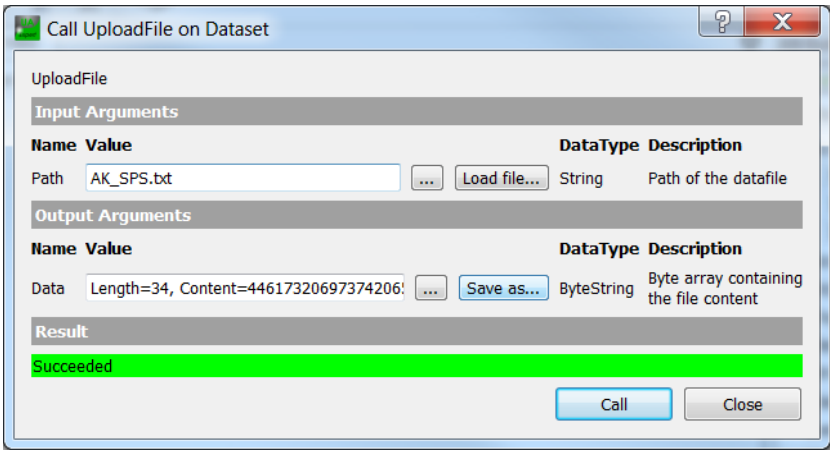


Path	<p>gibt den Ziel-Pfad auf der SPS samt dem File-Namen an</p> <p>Dieser Pfad muss im SPS-XML-File im Bereich Release eingetragen sein!</p> <p>Default: "C:\OPCUA\"</p> <p>Ohne Pfad-Angabe wird das erste im XML angegebene DIR verwendet.</p>
Data	<p>gibt die Quelle am Client an</p>

Ein Fehler deutet darauf hin, dass auf der SPS z.B. das OPC-UA-DIR fehlt.

Es kann nur in dieses DIR geschrieben werden!

9.6.2 File von Server (SPS) zu Client (Win)



Path	<p>gibt den Quell-Pfad auf der SPS samt dem File-Namen an</p> <p>Dieser Pfad muss im SPS-XML-File im Bereich Release eingetragen sein!</p> <p>Default: "C:\OPCUA\"</p> <p>Ohne Pfad-Angabe wird das erste im XML angegebene DIR verwendet.</p>
Data	<p>Das Data-Feld bleibt solange leer, bis der Button "Call" betätigt wird – dadurch wird das File zum Client übertragen, aber noch nicht gespeichert.</p> <p>Der Button "Save as" öffnet einen File-Browser zum Speichern in das Ziel-File.</p>

Ein Fehler deutet darauf hin, dass auf der SPS z.B. das OPC-UA-DIR fehlt.

Es kann nur aus diesem DIR gelesen werden!

10 Anhang A

	Featureliste Sigmatek OPC-UA Server				
Funktion	v02.01.005	v02.02.001	v02.03.001	V02.05.004	Kommentar
	Verbindung & Aufrufe				
Verbindung mehrerer Clients simultan	ja	ja	ja	ja	Bis zu 50 Sessions möglich
ReadNode	ja	ja	ja	ja	
WriteNode	ja	ja	ja	ja	
Subscriptions/MonitoredItems	ja	ja	ja	ja	
CallMethods Client=>Server	nur EM77	ja	ja	ja	
Auslösen von Events	nur EM77	ja	ja	ja	
	Verschlüsselung				
Verschlüsselte Verbindung via Basic256 S&E	ja	ja	ja	ja	
Verschlüsselte Verbindungen via Basic256Sha256 und Aes128Sha256Rsa	nein	nein	ja	ja	
Verschlüsselte Verbindungen via Aes256Sha256Rsa	nein	nein	nein	nein	
Erstellen von Self-Signed Certificates	ja	ja	ja	ja	
Zertifikatsverwaltung mit manuellem/auto-Trusting	ja	ja	ja	ja	
Ignorieren gewisser Zertifikatsfehler	nein	nein	ja	ja	Dies ist nicht für den Produktivbetrieb vorgesehen!
	Datentypen				
Basisdatentypen	ja	ja	ja	ja	
Arrays von Basisdatentypen	nein	ja	ja	ja	
Strukturen ohne Strings	ja	ja	ja	ja	
Arrays von Strukturen ohne Strings	nein	ja	ja	ja	
Strukturen mit Strings (fixe Länge)	nein	ja	ja	ja	
Arrays von Strukturen mit Strings (fixe Länge)	nein	nein	nein	nein	
Strukturen mit Strings	nein	nein	nein	nein	
Arrays von Strukturen mit Strings	nein	nein	nein	nein	
	Browsing				
BrowseNodes	ja	ja	ja	ja	
TranslateBrowsePathsToNodeIds	ja	ja	ja	ja	
	Weitere Services				
Userverwaltung mit Rollensystem	ja	ja	ja	ja	
Dateiübertragung	ja	ja	ja	ja	
Discovery GetEndpoints	ja	ja	ja	ja	
Historical Access Daten	nein	nein	ja	ja	
Historical Access Events	nur EM77	nur EM77	nur EM77	nur EM77	
Discovery	nein	nein	nein	nein	Konzeptphase
PubSub	nein	nein	nein	nein	Konzeptphase