BAUDIREKTION
AMT FÜR BETRIEB NATIONALSTRASSEN

DxNode OPC UA - Knoten Dokumentation

TECHNISCHE SPEZIFIKATIONEN NR. 09 FÜR BSA

Änderungsverzeichnis

Dok Name	Version	Datum	Verfasser	Bemerkung	Freigabe
Dokumentation v0.1 Light.docx	0.1	27.06.25	BECC	Ersterstellung	

Impressum

Dateiname:	Dokumentation v0.1 Light.docx
erstellt:	
Zielgruppe:	Unternehmer und Planer von BSA
genehmigt:	xx.xx.2025
Status:	Entwurf AfBN
Version:	0.1 vom 27.06.2025
Berichtstyp:	Referenzdokument
Titel:	TS 09 DxNodeUA Dokumentation
Autoren:	BECC

INHALTSVERZEICHNIS

1 Einführung 5
1.1 Allgemeines 5
1.2 Grundlagen 5
1.3 Abkürzungen 5
2 Übersicht 6
2.1 Einleitung 6
2.2 Allgemein 6
2.3 Das DxNode OPC UA-Netzwerk 7
3 Konzepte und Funktionen 10
3.1 Datenpunktliste 10
3.2 Verbindungsmatrix 10
4 Konfiguration 11
4.1 Hauptkonfiguration 11
4.2 OpcUA Server Konfiguration 12
4.3 Datenpunktliste 13
4.4 Verbindungsmatrix 13
4.5 Datenaustausch zwischen Knoten 15
4.6 Konfigurationsdownload 16
4.7 Logging 17
5 OPC UA-Addressspace 18
5.1 Datenpunkte 18
5.2 Alias 19
5.3 Verbindungsstatus 20
5.4 Übertragung von Alarmquittierungen 21
5.5 Fileserver 22
6 Verbindungssicherheit 24
6.1 Server Konfiguration 24
6.2 Client Konfiguration 25
7 Redundanz 27
8 Dateninitialisierung 28
9 Monitoring 29
10 Parametrierung 30
10.1 Datenpunktkonfiguration 31
10.2 Erzeugte Includefiles 32
11 Systemanforderung 38
12 Installation DxNode OPC UA 39
12.1 Kommandline Optionen 41
13 Anbindungsschnittstelle für BSA 42
14 Anwendungsbeispiel SCADA-Verbindung 43
14.1 Variable Lesen 44
14.2 Variable Schreiben 44
14.3 Variable Lesen und Schreiben 46
14.4 Variable mit Alarmquittierung 46

15 Darstellung des DxNode OPC UA Knotens im BLE und BSA HMI 48
15.1 Knotensymbol 48
16 Versionshistorie DxNode OPC UA 49
17 Zusatzinformationen DxNode Classic Knoten 50

1 Einführung

1.1 Allgemeines

Das Dokument beschreibt die Funktionsweise und Verwendung des DxNode OPC UA zum Datenaustausch in der Systemumgebung der GE XI zwischen der Betriebsleitebene (BLE) und der Betriebs- und sicherheitstechnischen Anlage (BSA).

Ziele des Dokumentes sind:

- Definition der Schnittstelle auf OPC UA Basis
- Grundlegende Funktion des DxNode OPC UA
- Kommunikation mit alten Anlagen (DxNode Classic)
- Konfiguration und Parametrierung
- Installation und Anbindung des DxNode OPC UA

1.2 Grundlagen

Folgende Dokumente bzw. Quellen wurden als Grundlage und als Anregungen für die Erstellung des vorliegenden Referenzdokumentes verwendet.

Titel/Inhalt	Verfasser
ASTRA 23001 Betriebs- und Sicherheitsausrüstungen (FHB BSA)	ASTRA
Unified Automation (https://www.unified-automation.com/)	Unified Automation
AfBN TS 07 BSA Handbuch	AfBN

1.3 Abkürzungen

Abkürzungen	Bedeutung
AKS	Anlagenkennzeichnungssystem
AR	Abschnittsrechner
BLE	Betriebsleitebene
BSA	Betriebs- und sicherheitstechnische Anlage
DxNode Classic (DxClassic)	Data Exchange Node (Datenaustauschknoten) Classic (Bezeichnung der alten Software auf Basis eines XML Protokolls) Vorgänger des DxNode OPC UA
DxNode OPC UA	Data Exchange Node (Datenaustauschknoten) Unified Architecture (Abgeleitet aus OPC UA) Softwarebibliothek, die die standardisierte Schnittstelle zwischen den BSA und zur BLE zur Verfügung stellt (früher DxNode oder TINWUR-Knoten).
DxNode OPC UA Netzwerk	Durch die Vernetzung aller DxNodeUA über das KOM-Netzwerk GE XI entsteht das DxNodeUA-Netzwerk (Früher DxNode oder TINWUR-Netzwerk).
Para-Tool	Zentrales Parametriertool
ProDx	Professional Data Exchange

2 Übersicht

2.1 Einleitung

Das vorliegende Dokument beschreibt die Funktionsweise und Verwendung des DxNode OPC UA. Die Schnittstelle basiert auf dem standardisierten OPC UA Protokoll und ist in der Gebietseinheit XI die Kommunikationsmiddleware zwischen BLE und BSA. Es richtet sich daher primär an Lieferanten von BSA-Anlagen.

2.1.1 Abgrenzung

Dieses Dokument beinhaltet nicht die Beschreibung und Verwendung des Engineering Tools „Paratool“ sowie einer Erklärung des OPC UA Standards. Der Anwender muss bereits grundlegende Kenntnisse über OPC UA haben.

2.2 Allgemein

Die DxNode OPC UA Schnittstelle stellt den Datenverkehr zwischen der BLE und den BSA-Anlagen, sowie auch den Datenverkehr zwischen den einzelnen BSA-Anlagen untereinander, mittels OPC UA sicher. Die Schnittstelle ist als Netzwerk-Knoten konzipiert. Zwischen zwei beliebigen DxNode OPC UA-Knoten kann eine standardisierte OPC UA Verbindung hergestellt werden, über die der gesamte Datenaustausch abläuft. Die Kommunikation erfolgt mittels standardisiertem OPC UA TCP Protokoll.

Das DxNode OPC UA Konzept basiert auf einem Software-Knoten (Node), ein standardisiertes Programm (auch Middleware genannt), das auf den einzelnen Rechnern der Betriebsleitebene und der BSA-Anlagen als komplette Funktionseinheit installiert wird. Alle DxNode OPC UA Knoten bilden zusammen das DxNode OPC UA-Netzwerk, das - ähnlich eines **Layers** im OSI-Modell - alle wesentlichen Funktionen des netzweiten Datenaustausches übernimmt.

Funktionen sind unter anderem:

- Automatischer Datenabgleich zwischen den DxNode OPC UA Knoten (Synchronisation)
- Speichern und Weiterleiten/Verteilen von Meldungen
- Ankopplung von bzw. an redundante Systeme
- Konfiguration für den Datenaustausch im Netzwerk inklusive aller anzubindenden Systeme
- Standardisierung der gesamten Kommunikation mittels OPC UA
- Erweiterbarkeit und Wartbarkeit des gesamten Systems
- Vereinfachung der individuellen Anbindungen (o.g. Funktionen sind bereits im DxNode OPC UA Knoten integriert)

Jeder DxNode OPC UA Knoten verwendet exakt die gleiche Software (EXE), die über XML-Konfigurationsdateien parametriert wird.

Vorteile:

- einheitliche systemweite standardisierte Schnittstelle
- Kommunikationsbeziehungen werden durch die Parametrierung festgelegt und sind von der Schnittstellenimplementierung unabhängig
- Zusätzliche Verbindungen (insbesonders Verbindungen für Reflexverbindungen zwischen individuellen BSA-Anlagen) sind leicht herstellbar, parametrierbar und bedürfen keiner Softwareänderung

Da die Umstellung auf DxNode OPC UA schrittweise zu erfolgen hat, besteht weiterhin die Möglichkeit, Daten mit dem DxNode Classic auszutauschen. Dies ist speziell für den Reflex- und Infotagdatenaustausch zwischen neuen BSA-Anlagen (die bereits mit DxNode OPC UA ausgestattet sind) und älteren BSA-Anlagen (welche noch mit dem DxNode Classic kommunizieren) erforderlich.

2.3 Das DxNode OPC UA-Netzwerk

Die Gesamtheit aller DxNode OPC UA-Knoten bilden das DxNode OPC UA-Netzwerk. Grundsätzlich tritt jeder DxNode OPC UA-Knoten als OPC UA-Server auf. Die Anbindung der BLE und die der lokale BSA erfolgt durch den nativen OPC UA-Client des jeweiligen verwendeten SCADA Produkts. Die Kommunikation zwischen den Knoten erfolgt mittels Knoteninternen OPC UA Clients entsprechend der jeweiligen Verbindungkonfiguration.

Das Konzept erlaubt den Ausbau zu einer standardisierten Netzwerk-Komponente mit erheblichen Vorteilen gegenüber anderen Lösungsansätzen:

- Konfigurierbare Datenpunktliste und Verbindungsmatrix
- Einfache Schnittstelle für Anbindungen durch standardisierte OPC UA Kommunikation
- Erweiterungen werden im gesamten DxNode OPC UA-Netzwerk wirksam

Die Erzeugung der Konfiguration aller Knoten des Netzwerkes erfolgt von einer zentralen Stelle, dem sogenannten PARA-Tool (Zuständigkeit UN BLE). Dieses erzeugt die für den einzelnen DxNode OPC UA-Knoten relevante Konfigurationsdateien in XML-Format. Diese werden vom jeweiligen Knoten bei Neustart automatisch von einer zentralen Stelle (über einen Fileserver siehe Kapitel 5.5) heruntergeladen.

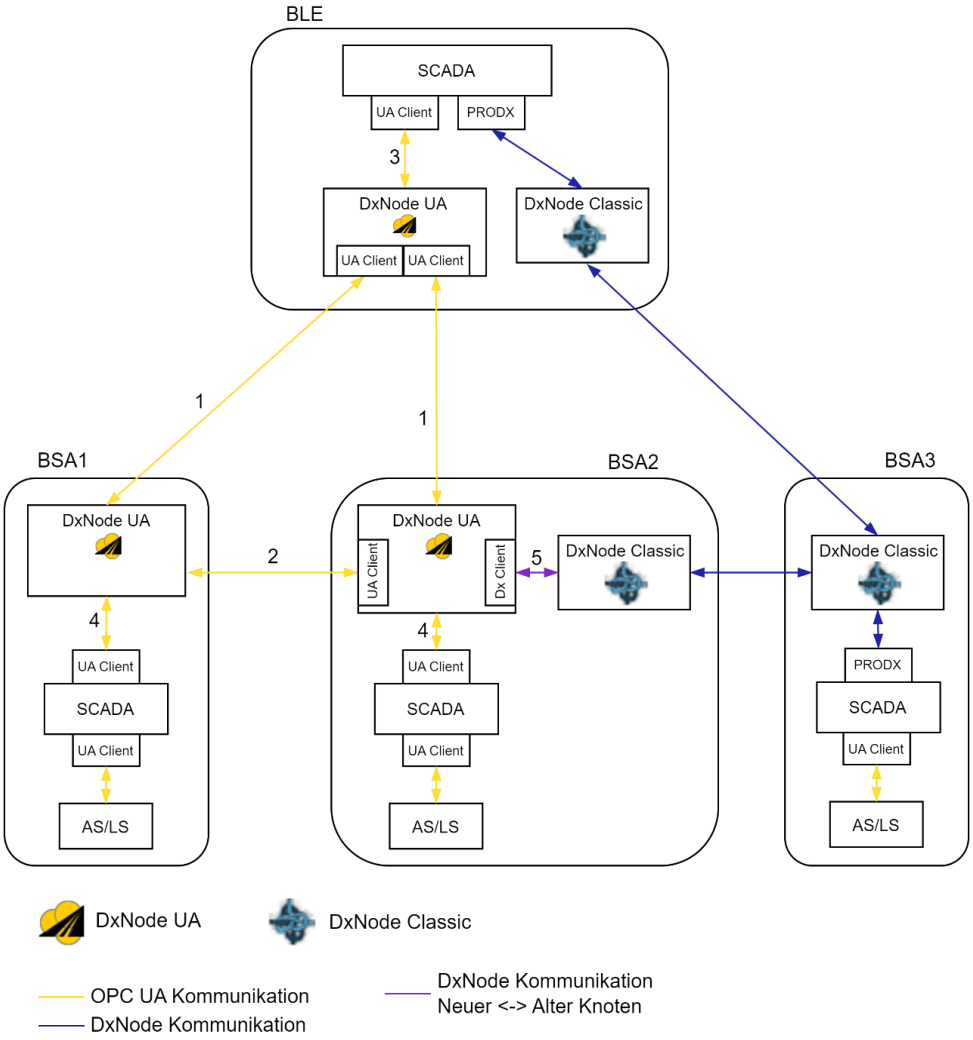


Abbildung 1: DxNode OPC UA Netzwerk

In einem DxNode OPC UA-Netzwerkes gibt es folgende Kommunikationspfade:

- zwischen Knoten <-> Knoten (Verbindungen 1, 2 und 5)
- zwischen BLE <-> Knoten (Verbindung 3)
- zwischen BSA <-> Knoten (Verbindung 4)

Die Kommunikationspfade basieren prinzipiell auf OPC UA unter Verwendung von OPC UA Data Access (Part 8 im OPC UA Standard) für den Austausch von Prozessdaten. Die einzige Ausnahme bildet die Anbindung an den DxNode Classic welches das proprietäre, XML basierende Protokoll des DxNode Classic verwendet.

Der generelle Datenaustauch entspricht der Vorgabe des Fachhandbuch BSA

Siehe Kapitel 6.1 „OPC UA“ -> ASTRA 23001 Betriebs- und Sicherheitsausrüstungen (FHB BSA)

Knoten – Knoten Kommunikation

Dieser Kommunikationspfad dient einerseits dem generellen Datenaustausch zwischen BLE-Knoten und BSA-Knoten (1), anderseits zwischen BSA-Knoten (2) zwecks Austausches von Info-Tags und Reflexen Klasse 2.

(1) BSA-Knoten sind OPC UA Server, der BLE-Knoten verbindet sich als OPC UA Client.

(2) BSA-Knoten der Senkenanlage ist OPC UA Server, der BSA-Knoten der Quellanlage verbindet sich als OPC UA Client.

BLE – Knoten Kommunikation

Verbindung zwischen BLE-Knoten und BLE (3). Der Knoten ist OPC UA Server und die BLE verbindet sich mittels OPC UA Client auf den Knoten.

BSA – Knoten Kommunikation

Verbindung zwischen BSA-Knoten und BSA (4). Der Knoten ist OPC UA Server und die BSA verbindet sich mittels OPC UA Client auf den Knoten.

Hinweis: Diese Kommunikationsschnittstelle ist die Verbindung, welche für den UN BSA relevant ist.

Knoten – Alter Knoten Kommunikation

Diese Verbindung (5) dient zum Austausch von Info-Tags und Reflexe Klasse 2 mit Bestandsanlagen, die noch nicht auf den neuen DxNode OPC UA umgestellt wurden. Der neue Knoten verbindet sich als Client auf den alten DxNode Classic.

Hinweis: Die DxNode Classic Applikation muss zusätzlich auf der BSA installiert sein. Das SCADA System bzw. BSA selbst kommuniziert ausschliesslich mit dem DxNode OPC UA!

3 Konzepte und Funktionen

3.1 Datenpunktliste

Ein DxNode OPC UA-Knoten besitzt ein lokales Prozessabbild mit den für ihn relevanten Datenpunkten. Diese werden in einer Datenpunktliste definiert. Grundsätzlich ist es möglich jeden Datenpunkt sowohl in einem lokalen als auch in einem globalen Adressraum darzustellen. Somit lassen sich proprietäre Adressen (lokale Datenpunktnamen) und systemweite AKS-Bezeichnungen (globale Datenpunktnamen) abbilden. Des Weiteren wird hier auch der Datentyp des jeweiligen Datenpunkts definiert.

3.2 Verbindungsmatrix

In der Verbindungsmatrix wird angegeben, wie die Werte der Datenpunkte mit den anderen Knoten im DxNode-Netzwerk auszutauschen sind.

Datenpunktliste und Verbindungsmatrix bilden grundsätzlich eine Einheit. Wenn ein Datenpunkt wegfällt/dazukommt, muss er in der Verbindungsmatrix gelöscht/hinzugefügt werden. Sonst können Inkonsistenzen entstehen.

Es können grundsätzlich zwei Arten von Verbindungen definiert werden: Aktive und passive Verbindungen.

Aktive Verbindung

Bei der aktiven Verbindung stellt der Knoten mittels OPC UA Client eine Verbindung zu einem anderen Knoten her. Innerhalb einer aktiven Verbindung wird definiert welche Datenpunkte auf welche Weise (lesend/schreiben, mit/ohne Alarmquittierung) mit dem anderen Knoten ausgetauscht werden.

Passive Verbindung

Die passive Verbindung dient lediglich der Verbindungsüberwachung, um anzuzeigen, ob eine Verbindung vom anderen Kommunikationspartner aufrecht ist. Innerhalb einer passiven Verbindung kann KEIN Datenpunktaustausch definiert werden.

Verbindungen von BSA-Anlagen und der BLE sind grundsätzlich passive Verbindungen. Die Konfiguration des Datenaustausches mit den Datenpunkten im Knoten obliegt dem jeweiligen SCADA-Tool (e.g. Peripherieadresse in WinCC OA) und kann nicht von der Knotenkonfiguration definiert werden.

Jede konfigurierte Verbindung (aktive sowie passive) wird mit internen Datenpunkten repräsentiert, die den aktuellen Verbindungszustand signalisieren (siehe Kapitel 5.3) entnehmen.

4 Konfiguration

Die Konfiguration des DxNode OPC UA besteht aus mindestens zwei XML basierten Konfigurationsfiles. Ein Hauptkonfigurationsfile und ein OPC UA Server Konfigurationsfile, welches vom Hauptkonfigurationsfile referenziert wird.

4.1 Hauptkonfiguration

Das Hauptkonfigurationsfile wird beim Start des Executables als Kommandlineoption (siehe Kapitel 12) mitgegeben wird und referenziert das OPC UA Server Konfigurationsfile. Datenpunktlisten und Verbindungsmatrizen können sowohl direkt im Hauptkonfigurationsfile definiert als auch über zusätzliche Files inkludiert werden.

```
<DxNodeUA>
  <Node prefix="DXUA_BSA2_" nn="ALL_NODE_BSA2"/>

  <!-- referenced opc ua server config file -->
  <UaServerConfig>OpcUaServerConfig.xml</UaServerConfig>

  <DPList>
    <!-- referenced dplist config files -->
    <include>DPListConfig_1.xml</include>
    <include>DPListConfig_2.xml</include>

    <!-- directly defined datapoints -->
    <Group gn="Group1">
      <P a="BSA_2.BsaWriteBleRead" n="+BSA2=BSAW_BLER" t="string"/>
      <P a="BSA_2.BleWriteBsaRead" n="+BSA2=BSAR_BLEW" t="string"/>
      <P a="BSA_2.BothWriteRead" n="+BSA1=BSABLERW" t="string"/>
    </Group>

    <P a="SpecialState" t="int32"/>
  </DPList>

  <ConnectionConfig>
    <!-- referenced connection config file -->
    <include>ConnectionsConfig.xml</include>

    <!-- directly defined active connection -->
    <Connect cn="CONN_BSA2_TO_BSA_1" url="opc.tcp://localhost:48011">
      <VarCon>
        <!--send data-->
        <P a="BSA_2.QuellReflex1" w="BSA_1.SenkReflex1"/>
        <P a="BSA_2.QuellReflex2" w="BSA_1.SenkReflex2"/>

      </VarCon>
    </Connect>

    <!-- directly defined passive connection -->
    <Connect cn="CONN_BLE_BSA_2"/>
  </ConnectionConfig>

</DxNodeUA>
```

Abbildung 2: Hauptkonfigurationsfile

Das referenzierte OPC UA Server Konfigurationsfile muss sich im gleichen `config` Ordner (oder relative dazu) befinden wie das Hauptkonfigurationsfile. Alle referenzierten Datenpunktlisten und Verbindungsmatrizen müssen in dem Ordner `include` abgelegt sein. Abbildung 3 zeigt ein Beispiel der Ordnerstruktur.

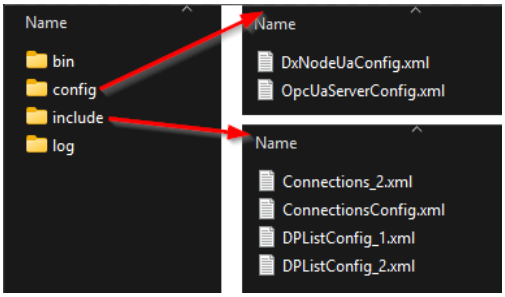


Abbildung 3: Ordnerstruktur Konfiguration

Der Tag `<Node>` beinhaltet die Attribute `prefix` und `nn` (NodeName). Der definierte Prefix wird OPC UA NodeIds, wie z.B. Variablen für Verbindungsinformationen, im OPC UA AddressSpace vorangestellt (siehe Kapitel 5.3).

4.2 OpcUA Server Konfiguration

Dieses Konfigurationsfile beinhaltet sämtliche Basiskonfiguration des OPC UA Servers. Unter anderem wird hier der Port festgelegt, über den sich ein OPC UA Client (z.B. BLE, BSA oder ein anderer DxNode OPC UA) auf den Server verbinden kann.

Das Format ist von dem zugrundeliegendem SDK von Unified Automation festgelegt. Abbildung 4 zeigt einen Auszug dieses Konfigurationsfiles. Weitere Details können der folgenden Dokumentation entnommen werden:

https://documentation.unified-automation.com/uasdkcpp/1.8.3/html/L2ServerSdkServerConfig.html#server_config_xml_file

```
<OpcServerConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <UaServerConfig>
        <!--Endpoint configuration
            This part of the configuration defines the OPC UA communication endpoints
            for the server and their security configurations-->
        <UaEndpoint>
            <SerializerType>Binary</SerializerType>
            <!-- URL of the Endpoint
                 This URL is used for Discovery and to open the Endpoints in the UA stack if no StackUrl is configured.
                 [NodeName] can be used as placeholder for the computer name. -->
            <Url>opc.tcp://[NodeName]:48010</Url>
            <!-- Optional URL that allows to define a specific address the stack should use to bind to.
                 Can be used to bind the endpoint to a specific network card or to localhost only.
            <StackUrl>opc.tcp://192.168.0.15:48010</StackUrl>
            -->
            <!-- It is up to an administrator to configure the actual exposed SecurityPolicies.
                 This SecurityPolicy should be disabled for most use cases.
            -->
            <SecuritySetting>
                <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#None</SecurityPolicy>
                <MessageSecurityMode>None</MessageSecurityMode>
            </SecuritySetting>
            <SecuritySetting>
                <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicy>
                <MessageSecurityMode>Sign</MessageSecurityMode>
                <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
            </SecuritySetting>
            <SecuritySetting>
                <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Aes128_Sha256_RsaOaep</SecurityPolicy>
                <MessageSecurityMode>Sign</MessageSecurityMode>
                <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
            </SecuritySetting>
            <SecuritySetting>
                <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Aes256_Sha256_RsaPss</SecurityPolicy>
                <MessageSecurityMode>Sign</MessageSecurityMode>
                <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
            </SecuritySetting>
            <!-- Flag indicating if the endpoint is provided in GetEndpoints and is therefore visible to a client. Default is true. -->
            <IsVisible>true</IsVisible>
            <!-- Flag indicating if the endpoint URL is provided as discovery URL. Default is true. -->
            <IsDiscoveryUrl>true</IsDiscoveryUrl>
            <!-- This option can be activated if certificates are used only for message security but not for application authentication.
                 If set to true, all client certificates will be accepted automatically and they are not stored.
                 It is strongly recommended to use this option only together with user authentication. -->
            <AutomaticallyTrustAllClientCertificates>false</AutomaticallyTrustAllClientCertificates>
```

Abbildung 4: Auszug OpcUaServer config

4.3 Datenpunktliste

Die Datenpunktliste definiert alle Datenpunkte mit lokalem Namen, Netzwerknamen und Datentyp die im Knoten für den weiteren Datenaustausch zu Verfügung stehen sollen. Diese Datenpunkte werden, wie im Kapitel 5 weiters erläutert, im Addressspace des OPC UA Server angelegt.

Abbildung 5 zeigt eine Beispielkonfiguration. Die Datenpunkte können einzeln oder zu Gruppen zusammengefasst definiert werde.

`a` Definiert den lokalen Namen der Variable

`n` Definiert den Netzwerknamen. Dieser ist optional und kann auch weggelassen werden.

`t` Definiert den Datentype der Variable (dies ist nötig da in OPC UA alle Variablen stark typisiert sind).

Folgende Datentypen sind unterstützt:

- `bool`
- `int32`
- `double`
- `string`
- `dynString`

```
<DxUaCnfExt>
    <DPList>
        <Group gn="GroupBefehl">
            <P a="BSA_1.BleWriteBsaRead" n="+BSA1=BSAR_BLEW" t="string"/>
            <P a="BSA_1.ReceiveInfo1" n="+BSA1=INFO1" t="string"/>
            <P a="BSA_1.SenkReflex1" n="+BSA1=SenkREFL1" t="string"/>
            <P a="BSA_1.SenkReflex2" n="+BSA1=SenkREFL2" t="bool"/>
        </Group>
        <Group gn="GroupMeldung">
            <P a="BSA_1.BsaWriteBleRead" n="+BSA1=BSAW_BLER" t="string"/>
            <P a="BSA_1.Status" n="+BSA1=STATU" t="int32"/>
        </Group>

        <P a="BSA_1.BothWriteRead" n="+BSA1=BSABLERW" t="string"/>
        <P a="SpecialState" t="int32"/>
    </DPList>
</DxUaCnfExt>
```

Abbildung 5: Beispiel Datenpunktliste

4.4 Verbindungsmatrix

Hier werden die Verbindungen definiert, die der Knoten zu anderen Knoten herstellen soll (aktive Verbindungen) und die Verbindungen, die der Knoten erwartet (passive Verbindungen). Des Weiteren wird pro aktive Verbindung definiert, wie und welche Datenpunkte mit dem anderen Knoten ausgetauscht werden sollen. Abbildung 6 zeigt eine Konfiguration mit aktiven und passiven Verbindungsdefinitionen.

```
<DxUaCnfExt>
    <ConnectionConfig>
        <Connect cn="CONN_BSA2_TO_BSA_1" type="DxUa" url="opc.tcp://localhost:48011">
            <VarCon>
                <!--send reflex-->
                <P a="BSA_2.QuellReflex1" w="BSA_1.SenkReflex1"/>
                <P a="BSA_2.QuellReflex2" w="BSA_1.SenkReflex2"/>
            </VarCon>
        </Connect>
        <Connect cn="CONN_BSA2_UA_TO_DX" type="DxClassic" host="127.0.0.1" port="8582">
            <VarCon>
                <!--send reflex/infotag-->
                <P a="BSA_2.QuellReflex2" w="="/>
                <P a="BSA_2.QuellInfotag1" w="="/>
                <!--receive reflex/infotag-->
                <P a="BSA_2.SenkReflex1" r="="/>
                <P a="BSA_2.SenkReflex2" r="="/>
                <P a="BSA_2.SenkInfotag1" r="="/>
            </VarCon>
        </Connect>
        <Connect cn="INTERN_1" type="DxIntern">
            <VarCon>
                <p a="BSA_2.ScadaConStat" r="WCCOAopcua:becc-mam-nb02:DxNodeUA_BSA:3.status"/>
            </VarCon>
        </Connect>
        <Connect cn="CONN_BLE_BSA_2"/>
        <Connect cn="WCCOAopcua:becc-mam-nb02:DxNodeUA_BSA:3"/>
    </ConnectionConfig>
</DxUaCnfExt>
```

Abbildung 6: Beispiel Verbindungsmatrix

Aktive Verbindung

Eine aktive Verbindungsdefinition beinhaltet einen Verbindungsnamen (`cn`) einen Typ (`type`) und typenspezifische Verbindungsinformationen. Folgende Verbindungstypen sind unterstützt:

- `DxUa`
- `DxClassic`
- `DxIntern`

DxUa

Definiert eine Verbindung zu einem anderen DxNode OPC UA. Ein interner OPC UA Client stellt mittels der angegebenen `url` die Verbindung zum anderen Knoten her.

DxClassic

Dieser Typ definiert eine Verbindung, um mit einem DxNode Classic Daten auszutauschen. Hier wird ein interner DxClassic Client erstellt, der sich mittels angegebenen `host` und `port` auf den klassischen DxNode verbindet.

DxIntern

Dieser Verbindungstype ermöglicht ein Knoteninternes mapping von Variablen.

Jede aktive Verbindung kann unter dem Tag `<VarCon>` definieren welche Variablenwerte in welche Richtung (lesend/schreibend) innerhalb dieser Verbindung ausgetauscht werden sollen. Details dazu sind im Kapitel 4.5 definiert.

Passive Verbindung

Eine passive Verbindungsdefinition beinhaltet nur einen Verbindungsnamen (`cn`). Diese Konfiguration wird rein für die Verbindungsüberwachung benötigt (siehe Kapitel 5.3).

4.5 Datenaustausch zwischen Knoten

Welche Daten und in welche Richtung (lesend und/oder schreibend) mit einem anderen Knoten ausgetauscht werden sollen, wird jeweils bei der aktiven Verbindungskonfiguration definiert (innerhalb des Tags `<VarCon>`, siehe Kapitel 4.4). Abbildung 7 zeigt hierzu einige Beispiele.

Jeder Tag `<P>` definiert einen Eintrag. Die Attribute `a` oder `n` geben den Namen der Variable an (entweder lokaler Name oder Netzwerkname) für die Daten mit dem anderen Knoten ausgetauscht werden soll.

Die Attribute `r` und `w` definieren, ob der Wert von anderen Knoten gelesen werden soll (`r`) und/oder zum anderen geschrieben werden soll (`w`). D.h. soll sowohl gelesen als auch geschrieben werden muss `r` und `w` angegeben werden. Mittels dieser Attribute muss der Name der Variable im anderen Knoten angegeben werden. Mit dem Eintrag `=` wird spezifiziert, dass der Name der gleiche ist wie im eigenen Knoten (lokal oder Netzwerkname, je nachdem, ob der Name mit `a` oder `n` angegeben wurde).

Lesen (`r`)

Es wird am anderen Knoten eine OPC UA Subscription erstellt und die MonitoredItems angelegt. Mittels derer Werteänderungen werden die Daten vom anderen Knoten eventgetrieben übermittelt.

Schreiben (`w`)

Jede Wertänderung wird zum anderen Knoten gesendet.

```
<P a="BSA_1.BsaWriteBleRead" r="="/>
<P a="BSA_1.BleWriteBsaRead" w="="/>
<P a="BSA_1.BothWriteRead" r="=" w="="/>

<!--send info-->
<P a="BLE.BleWriteInfo1" w="BSA_1.ReceiveInfo1"/>

<!--read reflex-->
<P a="BSA_1.SenkReflex1" r="="/>

<!--read with network name-->
<P n="+BSA1=NAME" r="="/>

<!--write with network name-->
<P n="+BSA1=NAMEBLEWRITE" w="="/>

<!-- reads status of bsa opcua client connection -->
<P a="BSA_1.ConnectionStatus" r="WCCOAopcua:becc-mam-nb02:DxNodeUA_BSA:2.status"/>
```

Abbildung 7: Beispiel Datenaustausch Konfiguration direkt

Anstatt einzelne Variablen anzugeben ist es auch möglich den Datenaustausch auf ganze Variablengruppen zu konfigurieren. Hierbei muss der Gruppenname (`gn`) angegeben, mit `a` oder `n` ein zusätzlicher Filter auf Lokalen- oder Netzwerknamen spezifiziert und mit `r/w` die Richtung des

Datenaustausches definiert werden. Der Character `*` im Filter definiert eine Wildcard. Im Gegensatz zu einer direkten Verknüpfung ist bei den Attributen `r` und `w` nur der Eintrag `=` zulässig.

```
<VarCon>
    <!--connection via group -->
    <P gn="GroupBefehl" n="+*" w="="/>
    <P gn="GroupMeldung" n="+*" r="="/>
</VarCon
```

Abbildung 8: Beispiel Datenaustausch Konfiguration mittels Gruppen

4.6 Konfigurationsdownload

Der DxNode OPC UA besitzt die Möglichkeit Datenpunktlisten und Verbindungsmatrizen, die im Hauptkonfigurationsfile mittels `<include>` eingebunden sind, bei Neustart des Knoten von einem zentralen DxNode OPC UA, der als Fileserver dient, mittels OPC UA Methoden zu beziehen beziehungsweise abzugleichen. Hierzu muss das Hauptkonfigurationsfile um den Tag `<FileServer>` erweitert werden. Dieser spezifiziert mittels dem Attribute `url`, unter Angabe eines Ports, die Verbindung zu dem zu benutzenden Fileserver-Knoten. Die `<SecuritySettings>` sind optional und abhängig von den konfigurierten Verbindungssicherheiten am Fileserver-Knoten (siehe Kapitel 6). Eine detaillierte Beschreibung der `<SecuritySettings>` ist dem Kapitel 6.2 zu entnehmen. Die Funktionalität Fileserver ist im Kapitel 5.5 beschrieben.

```
<DxNodeUA>
  <Node prefix="DXUA_BSA2_" nn="ALL_NODE_BSA2"/>

  <!-- referenced opc ua server config file -->
  <UaServerConfig>OpcUaServerConfig.xml</UaServerConfig>

  <!-- defines access to the filesever-->
  <FileServer url="opc.tcp://localhost:48010">
    <SecuritySetting>
        <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicy>
        <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>

        <ClientCertificate>pkiclient/own/certs/dxnodeuaclient.der</ClientCertificate>
        <ClientPrivateKey>pkiclient/own/private/dxnodeuaclient.pem</ClientPrivateKey>
    </SecuritySetting>
  </FileServer>

  <DPList>
    <!-- referenced dplist config files -->
    <include>DPListConfig_1.xml</include>
    <include>DPListConfig_2.xml</include>
```

Abbildung 9: FileServer config

Bei jedem Neustart wird überprüft, ob sich referenzierte Includefiles am Fileserver-Knoten gegenüber den Files im lokalen `include` Folder geändert haben. Bei detektierter Änderung oder falls das File lokal noch nicht vorhanden ist, wird es in den lokalen Folder kopiert. Files, die nur lokal existieren, nicht aber am Netzlaufwerk, werden vom Abgleichmechanismus ignoriert und so belassen wie sie sind.

Nach erfolgtem Abgleich werden in weiterer Folge ausschliesslich die lokalen Includefiles herangezogen. Dies gilt auch für den Fall, dass bei Neustart der konfigurierte Fileserver-Knoten nicht erreichbar ist.

Ist kein `<FileServer>` definiert, startet der DxNode OPC UA regulär mit den lokalen Includefiles.

4.7 Logging

Der DxNode OPC UA generiert Logmessages sowohl direkt in der Konsole als auch in einem Logfile. Die angewandten Loglevels können für beide Ausgaben getrennt konfiguriert werden. Dafür stehen folgende Levels zur Verfügung:

- `FATAL`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`

Die Logfiles werden unter dem Folder `log/` mit dem Namen `DxNodeUa.log` abgelegt. Das Filelogging ist als RollingFileAppender ausgeführt. Sowohl die maximale Filegrösse (in MB) als auch die Anzahl der Backupfiles kann konfiguriert werden (siehe Abbildung 11).

Abbildung 10 zeigt den optionalen Log-Konfigurationseintrag im Hauptkonfigurationsfile. Die gezeigten Werte sind zugleich die Standardwerte, falls der Eintrag nicht vorhanden ist.

```
<DxNodeUA>
  <Node prefix="DXUA_BSA2_" nn="ALL_NODE_BSA2"/>

  <!-- referenced opc ua server config file -->
  <UaServerConfig>OpcUaServerConfig.xml</UaServerConfig>

  <!-- defines access to the filesever-->
  <FileServer url="opc.tcp://localhost:48010">

  <!-- defines the logging configuration -->
  <LogConfig>
    <ConsoleLogLevel>INFO</ConsoleLogLevel>
    <FileLogLevel>INFO</FileLogLevel>
    <MaxFileSize>10</MaxFileSize>
    <MaxBackupFile>1</MaxBackupFile>
  </LogConfig>

  <DPList>
```

Abbildung 10: Log Konfiguration

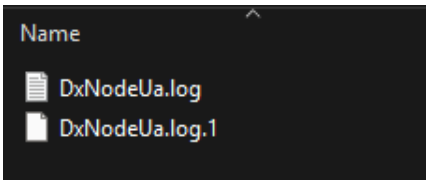


Abbildung 11: Log Files

5 OPC UA-Addressspace

5.1 Datenpunkte

Alle Datenpunkte, die in der Datenpunktliste definiert sind, werden im OPC UA Server Addressspace als BaseDataVariable angelegt.

Die NodeId befindet sich im Namespace 2 und verwendet den lokalen Namen als string Identifier. D.h. für die Variable mit dem lokalen Namen `BSA_1.SenkReflex1` ergibt sich die NodeId: `ns=2;s=BSA_1.SenkReflex1`.

Alle Variablen werden im OPC UA-Addressspace organisatorisch unter dem Ordner `DxNode` eingeordnet. Ist der lokale Name mit Punkten getrennt (so wie im oben genannten Beispiel), werden weitere Unterordner erstellt. Als DisplayName wird immer der jeweilige Teilabschnitt verwendet. Abbildung 12 zeigt dazu ein Beispiel.

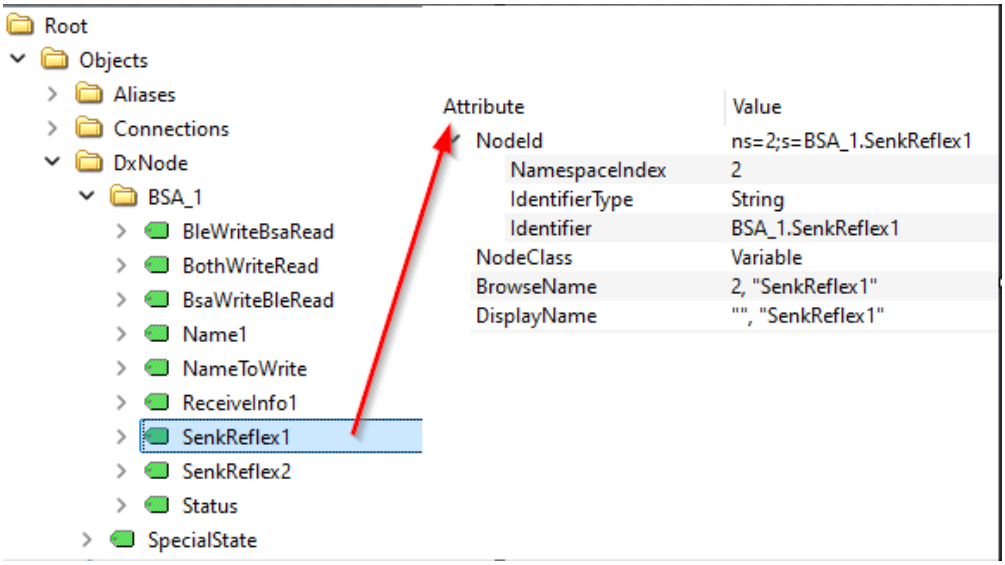


Abbildung 12: Beispiel Datenpunkte im OPC UA Addressspace

Tabelle 1 listet das Mapping zwischen den Datentypen in der XML Datenpunktliste und den resultierenden OPC UA Datentypen.

Tabelle 1: DataType mapping

DataType in XML	OPC UA DataType
`bool`	`Boolean`
`int32`	`Int32`
`double`	`Double`
`string`	`String`
`dynString`	`String Array`

5.2 Alias

Um, den von DxClassic definierten Netzwerknamen (auch als globaler Name bezeichnet) abzubilden, wird in OPC UA das Alias Feature verwendet (siehe Abbildung 13).

Die eigentliche NodeId des Datenpunkts, die sich aus dem definierten lokalen Namen zusammensetzt (siehe Kapitel 5.1) und für den Datenaustausch benötigt wird, kann vom Server mittels der `FindAlias` Methode abgefragt werden wie in Abbildung 14 ersichtlich ist.

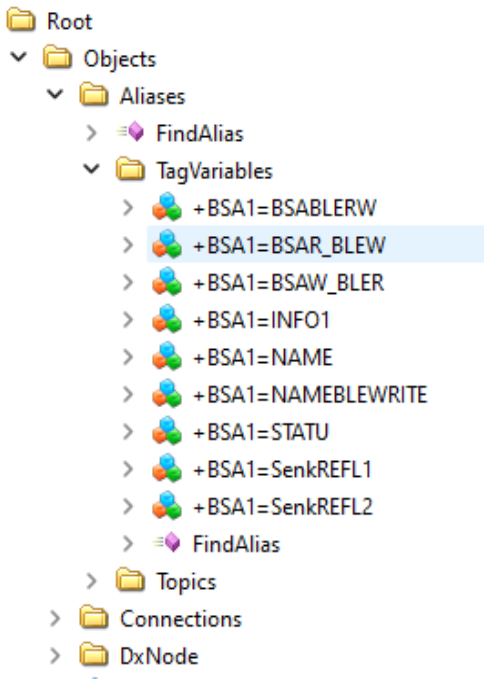


Abbildung 13: Beispiel Alias (globaler Name)

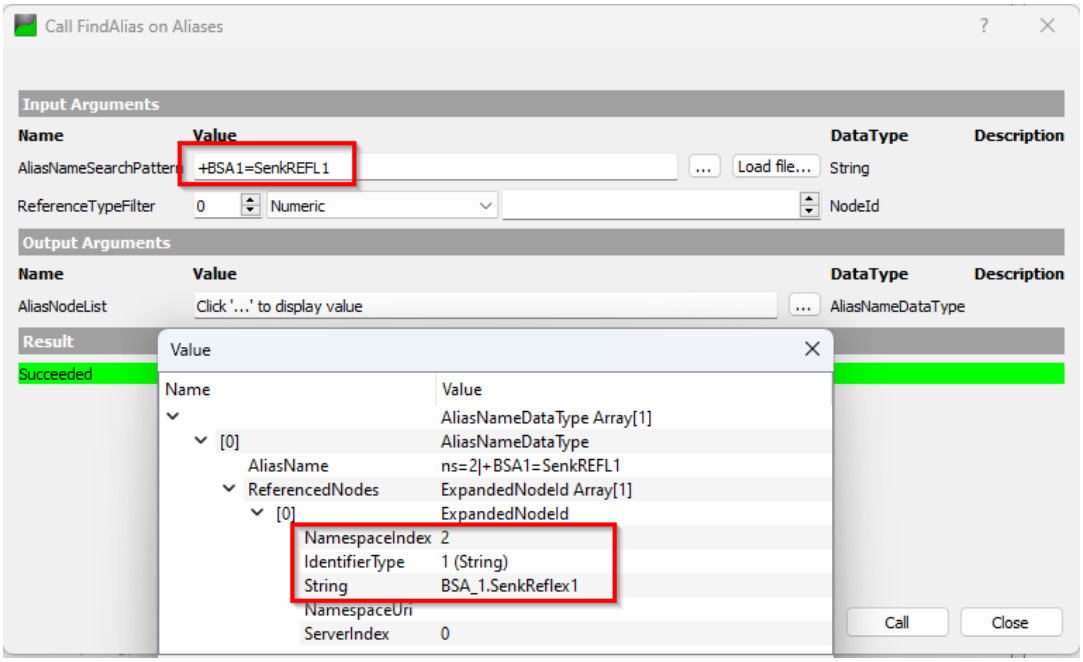


Abbildung 14: Alias Abfrage

5.3 Verbindungsstatus

Für jede definierte Verbindung (aktive und passiv) wird unter der Ordnerstruktur `Connections/ConnectionName` die DataVariable *prefix*+*ConnectionName*`.status` im Namespace 2 erstellt (siehe Abbildung 15).

Mittels dieser Variable wird der Verbindungsstatus der jeweiligen Verbindung angezeigt. Folgende Werte sind möglich:

- `0` Nicht verbunden
- `1` Verbindung wird aufgebaut (nur bei aktiven Verbindungen)
- `2` Verbindung hergestellt.

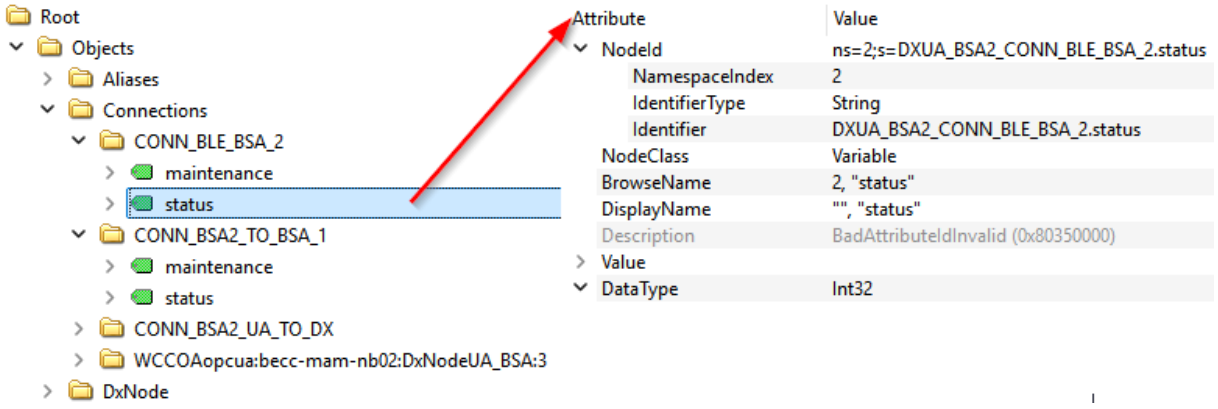


Abbildung 15: Verbindungsstatus

Das Maintenanceflag erlaubt es, die Verlustmeldung einer Verbindung zu unterdrücken. Solange `maintenance` auf `true` gesetzt ist, wird `status` dieser Verbindung immer auf `2` gesetzt, unabhängig vom tatsächlichen Verbindungszustand mit dem anderen Knoten. Bei (Rück)setzen von `maintenance` auf `false` wird mit `status` umgehend der tatsächliche Verbindungszustand wiedergegeben.

Das Maintenanceflag ist bei Knotenneustart defaultmässig auf `false` gesetzt.

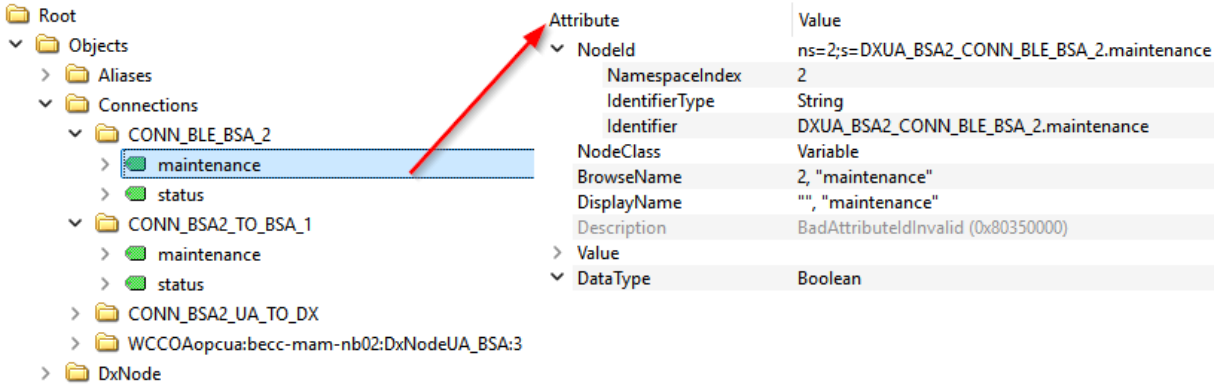


Abbildung 16: Verbindung Maintenanceflag

5.4 Übertragung von Alarmquittierungen

Wenn für einen Datenpunkt die Quittier-Information ausgetauscht werden soll, muss dies in der Verbindungsdefinition explizit mittels dem Attribut `attr="A"` konfiguriert werden.

```
<Connect cn="CONN_BLE_BSA_1" url="opc.tcp://localhost:48011">
    <VarCon>
        <P a="BSA_1.BsaWriteBleRead" r="="/>
        <P a="BSA_1.BleWriteBsaRead" w="="/>
        <P a="BSA_1.BothWriteRead" r="=" w="="/>

        <P n="+BSA1=STATU" r="=" attr="A"/> <!-- read status via network name with up/down ack for alerts -->

    <VarCon>
</Connect>
```

Abbildung 17: Verbindungskonfiguration mit Alarmquittierung

Hierzu wird in beiden Knoten ein zusätzlicher boolscher Datenpunkt mit dem Namen *DpName*`.alertAck` erstellt und diese mit `rw` verbunden.

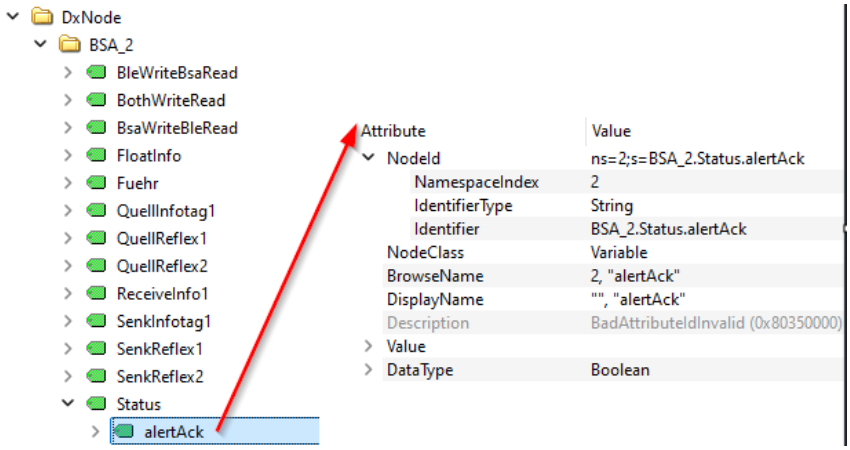


Abbildung 18: AlertAck Datenpunkt

Dieser zusätzliche Datenpunkt kann dann vom jeweiligen SCADA System verwendet werden, um Alarmquittierungen zu signalisieren oder zu empfangen.

Quittierung signalisieren

Um zu signalisieren, dass quittiert wurde, muss das SCADA System die `alertAck` Variable auf `true` setzen. Der Wert wird vom Knoten automatisch nach einer Sekunde wieder auf `false` gesetzt.

Quittierung empfangen

Wird vom SCADA System eine steigende Flanke detektiert (Wert von `alertAck` geht von `false` auf `true`), ist dies ein Anzeichen, dass im anderen SDADA System der Alarm quittiert wurde und der eigene Alarm ebenfalls quittiert, werden soll.

Die generelle Funktion der Quittierung entspricht der Vorgabe des Fachhandbuch BSA

Siehe Kapitel 12.3.6.2 „Ohne OPC UA AC“ -> ASTRA 23001 Betriebs- und Sicherheitsausrüstungen (FHB BSA)

Diese Mechanismen müssen in den jeweiligen SCADA Systemen implementiert werden und liegen ausserhalb der Zuständigkeit von DxNode OPC UA. DxNode OPC UA sorgt lediglich für den Transport der Quittier Information mittels der `alertAck` Variable. Im Falle von WinCC OA kann dies in einem ControlScript implementiert werden.

Zu beachten gilt, dass die Quittierinformation nur bei aufrechter Verbindung aller im Pfad beteiligter Komponenten übertragen wird.

5.5 Fileserver

Ein DxNode OPC UA kann für andere Knoten als Fileserver dienen, um als zentrale Stelle Files (z.B. Includesfiles, siehe Kapitel 4.6) zum Download zur Verfügung zu stellen. Dazu verfügt der Knoten über die zwei OPC UA Methoden `HasFile()` und `GetFile()` (siehe Abbildung 19) die von jeweiligem Client aufgerufen werden können. Alle Files, die über diesen Mechanismus verfügbar sein sollen, müssen sich im Folder `fileServer/` befinden (siehe Abbildung 20).

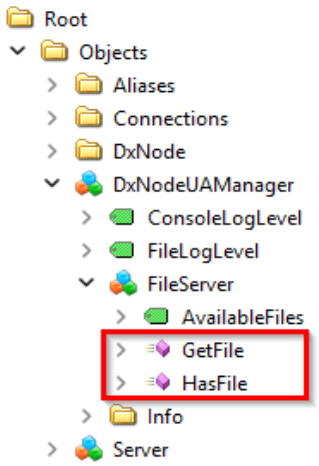


Abbildung 19: FileServer Methoden im UA Addressspace

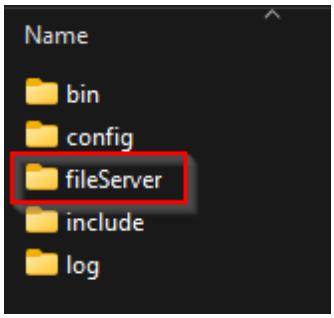


Abbildung 20: Ordnerstruktur mit fileServer Folder

Abbildung 21 zeigt den Beispielaufruf der `GetFile()` Methode.

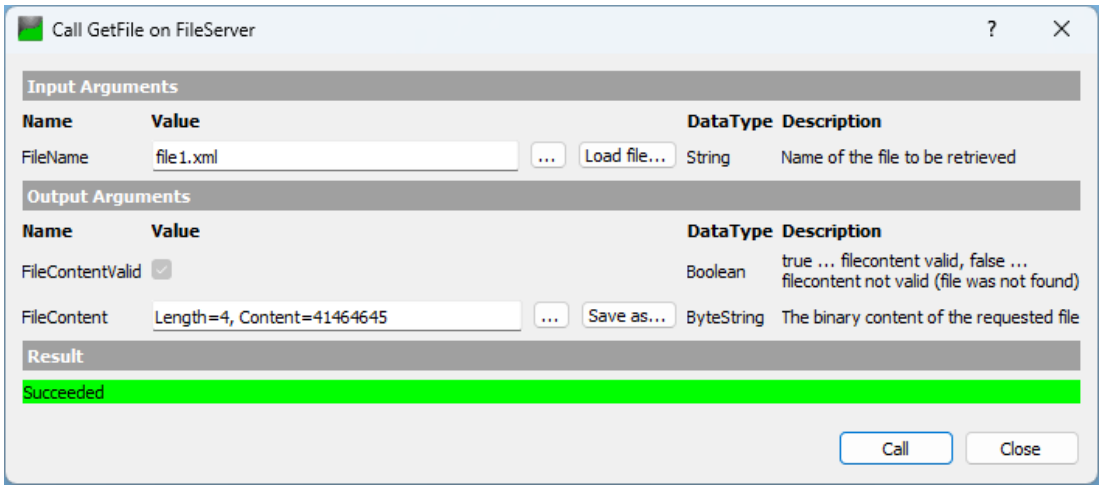


Abbildung 21: Beispiel Aufruf GetFile()

6 Verbindungssicherheit

Die OPC UA Verbindungen können mit den im Standard definierten Mechanismen abgesichert werden, um unbefugten Kommunikationsaufbau zu unterbinden und eine etwaige Datenmanipulation zu verhindern. Die Absicherung erfolgt dabei auf Zertifikatsbasierten private/public Keys. Dabei sind folgende Security Policies und Security Modes möglich:

Policies:

- `None`
- `Basic256Sha256`
- `Aes128_Sha256_RsaOaep`
- `Aes256_Sha256_RsaPss`

Modes:

- `None`
- `Sign`
- `SignAndEncryp`

6.1 Server Konfiguration

Im `OpcUaServerConfig.xml` in Sektion `<UaEndpoint>` wird definiert, welche Verbindungsmethoden der OpcUA Server des DxNode OPC UA akzeptiert (siehe Abbildung 22).

```
<SecuritySetting>
    <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#None</SecurityPolicy>
    <MessageSecurityMode>None</MessageSecurityMode>
</SecuritySetting>
<SecuritySetting>
    <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicy>
    <MessageSecurityMode>Sign</MessageSecurityMode>
    <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
</SecuritySetting>
<SecuritySetting>
    <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Aes128_Sha256_RsaOaep</SecurityPolicy>
    <MessageSecurityMode>Sign</MessageSecurityMode>
    <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
</SecuritySetting>
<SecuritySetting>
    <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Aes256_Sha256_RsaPss</SecurityPolicy>
    <MessageSecurityMode>Sign</MessageSecurityMode>
    <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
</SecuritySetting>
```

Abbildung 22: Auszug SecuritySettings OpcUaServerConfig

Die Zertifikate, die der Server benutzt, werden in einer Folder Struktur unter dem Folder `pkiserver` abgelegt (siehe Abbildung 23).

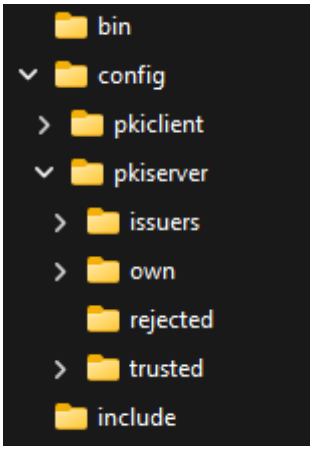


Abbildung 23: PKI Server

Das eigene Zertifikat liegt in `own/certs` mit dem Standardnamen `dxnodeuaserv.der` und der dazugehörige private Key in `own/private` mit dem Standardnamen `dxnodeuaserv.pem`. Die Zertifikate der zugelassenen Clients liegen unter `trusted/certs`. Verbindet sich ein Client, dessen Zertifikat nicht bekannt ist (nicht im `trusted/certs` vorhanden) wird dieses unter `rejected` abgelegt und die Verbindung nicht zugelassen.

Die Namen des eigenen Zertifikats und private Keys kann im `OpcUaServerConfig.xml` in der Sektion `<ServerCertificate>` konfiguriert werden.

```
<OpenSSLStore>
    <ServerCertificate>[ConfigPath]/pkiserver/own/certs/dxnodeuaserv.der</ServerCertificate>
    <ServerPrivateKey>[ConfigPath]/pkiserver/own/private/dxnodeuaserv.pem</ServerPrivateKey>
</OpenSSLStore>
```

Abbildung 24: Server Zertifikat Konfiguration

Hinweis:

Die SecurityPolicy wird Standardkonform als URL angegeben, es wird aber zur Laufzeit keine Internetverbindung benötigt!

6.2 Client Konfiguration

Bei jeder aktiven Verbindung vom Typ `DxUa` (siehe Kapitel 4.4) und der Verbindung für `FileDownload` (siehe Kapitel 4.6) kann optional definiert werden, welchen Security Settings der Client verwenden soll, um sich mit dem Server zu verbinden. Standardmässig (wenn nichts explizit konfiguriert ist) wird für Policy und Mode `None` verwendet.

```
<Connect cn="CONN_BSA2_TO_BSA_1" type="DxUa" url="opc.tcp://localhost:48011">
    <SecuritySetting>
        <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicy>
        <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
    </SecuritySetting>
    <VarCon>
        <!--send reflex-->
        <P a="BSA_2.QuellReflex1" w="BSA_1.SenkReflex1"/>
        <P a="BSA_2.QuellReflex2" w="BSA_1.SenkReflex2"/>
    </VarCon>
</Connect>
```

Abbildung 25: Verbindungskonfiguration mit Security Settings

Die Zertifikate, die von den Clients benutzt werden, liegen in einer Folder Struktur unter dem Folder `pkiclient` (siehe Abbildung 26).

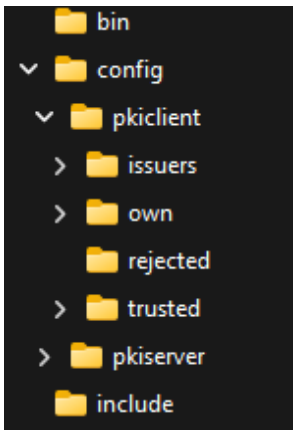


Abbildung 26: PKI Client

Das eigene Zertifikat liegt in `own/certs` mit dem Standardnamen `dxnodeuaclient.der` und der dazugehörige private Key in `own/private` mit dem Standardnamen `dxnodeuaclient.pem`. Die Zertifikate der vertrauenswürden Servers liegen unter `trusted/certs`. Bekommt der Client beim Verbindungsaufbau vom Server ein Zertifikat, dass ihm nicht bekannt ist (nicht im `trusted/certs` vorhanden) wird dieses unter `rejected` abgelegt und der Verbindungsaufbau abgebrochen.

Die Namen des eigenen Zertifikats und private Keys kann optional pro aktive Verbindung in der Sektion `<SecuritySettings>` konfiguriert werden. Ist die Konfiguration nicht vorhanden werden die Standardnamen verwendet.

```
<Connect cn="CONN_BSA2_TO_BSA_1" type="DxUa" url="opc.tcp://localhost:48011">
    <SecuritySetting>
        <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicy>
        <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>

        <ClientCertificate>pkiclient/own/certs/dxnodeuaclient.der</ClientCertificate>
        <ClientPrivateKey>pkiclient/own/private/dxnodeuaclient.pem</ClientPrivateKey>
    </SecuritySetting>
```

Abbildung 27: Verbindungskonfiguration mit Zertifikatsoptionen

7 Redundanz

Der DxNode OPC UA ist darauf ausgelegt eine vollständige Hot-Standby Redundanz zu ermöglichen. Abbildung 28 zeigt dieses Szenario anhand eines Beispiels mit redundanter BLE und einer redundanten BSA inklusive den darin involvierten OPC UA Verbindungen.

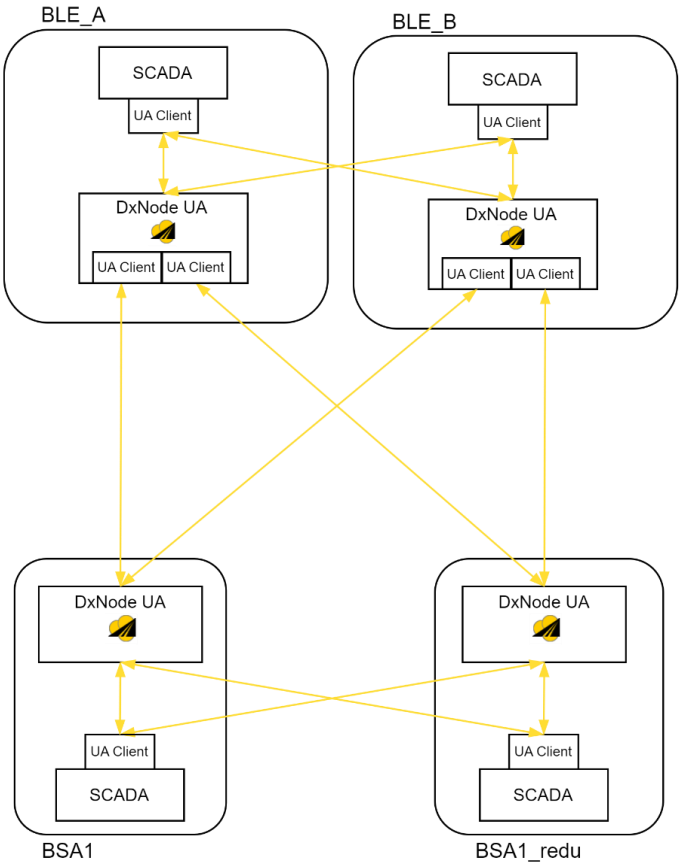


Abbildung 28: Redundanter Aufbau

8 Dateninitialisierung

Nach Verbindungsaufbau zwischen zwei Knoten werden anhand der konfigurierten Datenaustauschdefinitionen bei den aktiven Verbindungsdefinitionen (siehe Kapitel 4.5), die Werte zwischen den Knoten synchronisiert.

Werte, die von einer BSA oder von der BLE vorgegeben werden (vom jeweiligen SCADA System zum Knoten geschrieben), müssen nach Verbindungsaufbau in den Knoten geschrieben werden. Dies kann z.B. durch eine Generalabfrage ausgelöst werden.

9 Monitoring

Um Werteänderungen im DxNode OPC UA-Netzwerk z. B. für Debugzwecke zu beobachten, kann prinzipiell jedes OPC UA Client Programm verwendet werden, da DxNode OPC UA auf dem OPC UA Standard basiert. Ein weitverbreitetes, frei erhältliches Tool ist UaExpert von Unified Automation, wie Abbildung 29 zeigt.

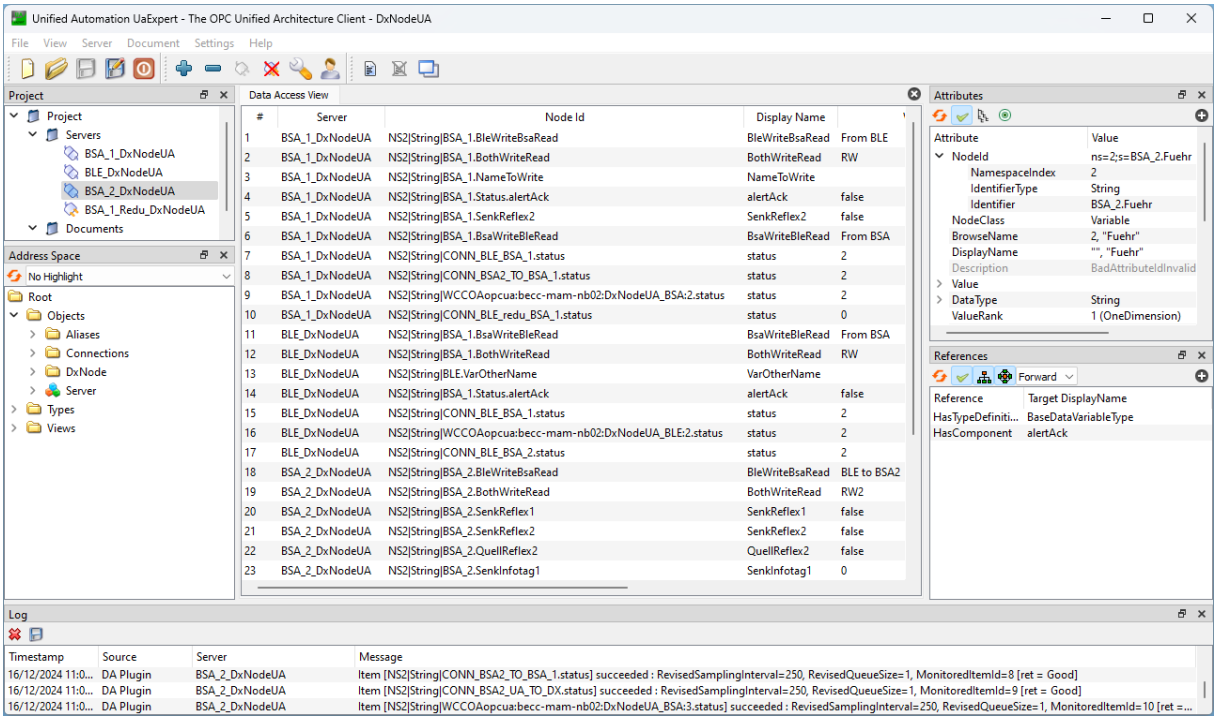


Abbildung 29: Unified Automation UaExpert

10 Parametrierung

Die Includefiles für Datenpunktlisten und Verbindungen werden für die einzelnen Knoten in der Regel über das zentrale Konfigurationstool, das ParaTool, erzeugt. Beim Konfigurieren einer BSA muss definiert werden, über welchen Knotentyp die Anlage ins System angebunden ist.

Folgende Typen sind möglich:

- `DxNode Classic` Anlagen die mit dem DxNode Classic angebunden sind.
- `DxNode UA` Anlagen die mit dem DxNode OPC UA angebunden sind.
- `DxNode UA+Classic` Anlagen die mit dem DxNode OPC UA angebunden sind, aber Reflexe und Infotags mit Anlagen mit DxNode Classic austauschen müssen.

Bei Anlagen mit DxNode OPC UA wird die URL des OpcUA Servers mit dem Feld `OpcUa URL` definiert.

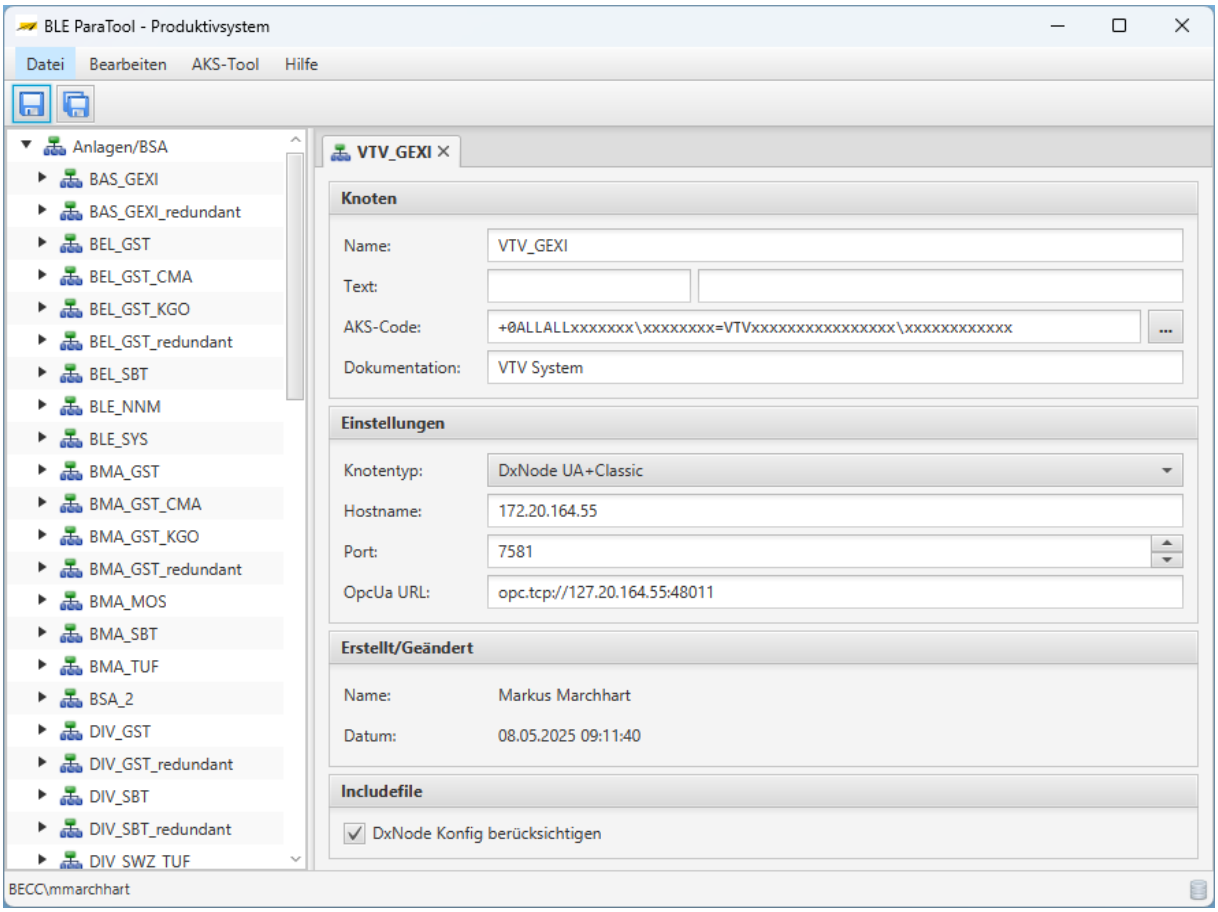


Abbildung 30:ParaTool Anlagenkonfiguration

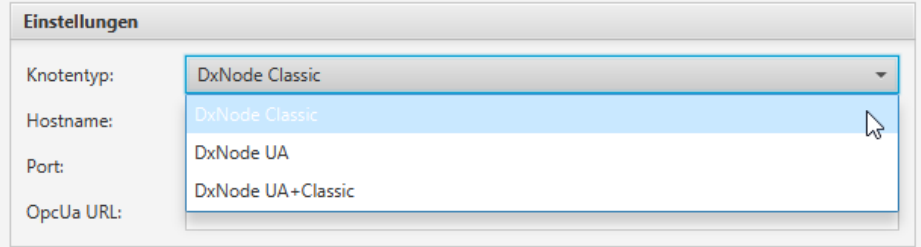


Abbildung 31: Knotentypauswahl

10.1 Datenpunktkonfiguration

Jeder Datenpunkt, für den Werte über das DxNode OPC UA-Netzwerk ausgetauscht werden sollen, muss im ParaTool das `DxNode` config besitzen. Dies definiert, wie der Wert zwischen BLE und BSA auszutauschen ist (lesen und/oder schreiben), ob Quittier-Information übermittelt werden soll sowie einen optionalen alternativen lokalen Namen. Im BLE-Knoten setzt sich der lokale Name des Datenpunkts aus der Struktur im ParaTool zusammen. Für den lokalen Namen des Datenpunkts im BSA-Knoten wird entweder der alternative konfigurierte lokal Name herangezogen oder setzt sich ebenfalls aus der Struktur im ParaTool zusammen. Als Netzwerkname wird der konfigurierte AKS des Datenpunkts herangezogen. Beispiel siehe Abbildung 32, Abbildung 33 und Abbildung 34.

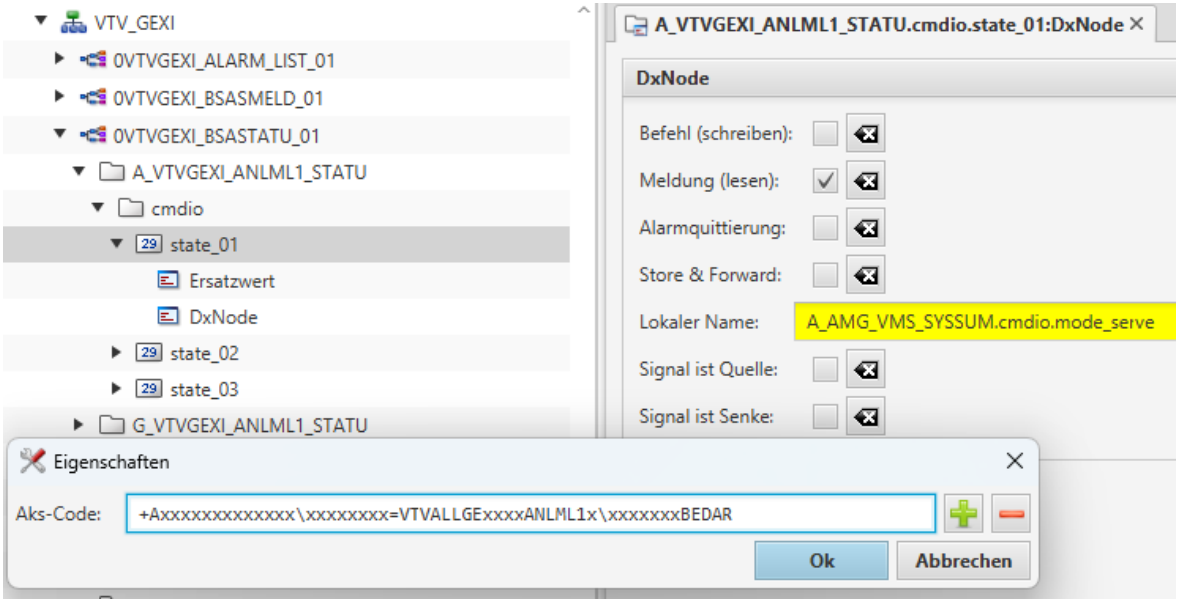


Abbildung 32: ParaTool DxNode config

Tabelle 2: DxNode Config Optionen

Option	Beschreibung
`Befehl (schreiben)`	Der Wert wird von der BLE geschrieben und von der BSA gelesen. Signalfluss BLE -> BSA
`Meldung (lesen)`	Der Wert wird von der BLE gelesen und von der BSA geschrieben. Signalfluss BSA -> BLE

`Alarmquittierung`	Quittier Information wird in beide Richtungen ausgetauscht. Dazu muss mindestens `Befehl` oder `Meldung` deklariert sein.
`Store & Forward`	Hat für DxNode OPC UA keine Bedeutung.
`Lokaler Name`	Optionaler alternativer lokaler Name mit dem der Datenpunkt im BSA-Knoten erstellt werden soll. Ist nichts angegeben wird der Name entsprechend dem Namen im ParaTool herangezogen.
`Signal ist Quelle`	Relevant, wenn der Datenpunkt einen Quellreflex oder Infotag darstellt und die BSA diesen mittels DxClassic an andere BSAs sendet.
`Signal ist Senke`	Relevant, wenn der Datenpunkt einen Senkenreflex oder Infotag darstellt und die BSA diesen mittels DxClassic von anderen BSAs empfängt.

Ist `Befehl` und `Meldung` selektiert, können sowohl BLE als auch BSA lesend und schreiben auf den Datenpunkt zugreifen (bidirektionaler Signalfluss).

10.2 Erzeugte Includefiles

Das ParaTool erzeugt an zwei Stellen Includefiles für die Knoten des DxNode OPC UA-Netzwerks. Einerseits beim Aktivieren einer Version der Objektinstanzen einer BSA, die über DxNode OPC UA angebunden ist, andererseits bei der Versionsaktivierung der Kommunikationsmatrix einer BSA, die über DxNode OPC UA angebunden ist.

Hinweis: Alle weiteren Erklärungen in diesem Kapitel (falls nicht anders angegeben) betreffen immer Anlagen, die mit DxNode OPC UA angebunden sind!

Bei Anlagen, die über DxNode OPC UA angebunden sind und zusätzliche eine DxClassic zwecks Reflexaustausch besitzen werden auch, wie bisher, alle benötigten Files für den DxClassic erzeugt!

Objektinstanzen

Tabelle 3 zeigt alle Includefiles, die bei der Aktivierung einer Objektinstanzenversion einer BSA erstellt werden.

Tabelle 3: Erzeugte Includefiles Objektinstanzen

Includefile	Beschreibung
`ALL_DPUA_BLE.xml`	Beinhaltet alle Datenpunkte aller BSAs für den BLE-Knoten.
`ALL_MXUA_BLE_A.xml` `ALL_MXUA_BLE_B.xml`	Beinhaltet die aktiven Verbindungsdefinitionen für den BLE-Knoten zu den BSA-Knoten.
`DPUA_`*`BSAID`*`.xml`	Beinhaltet die Datenpunkte der einzelnen BSA für deren Knoten.
`CXUACL_`*`BSAID`*`.xml`	Ein optionales File welches erzeugt wird, falls die BSA mit DxClassic Anlagen Reflexe und Infotags austauschen muss. Dies beinhaltet die aktive Verbindungsdefinition für den BSA-Knoten zum DxClassic Knoten.

Die BSAID für die Filenamen setzt sich aus den AKS-Teilen Erfassungsbereich, Streckenabschnitt, Anlage und Anschluss des AKS der BSA zusammen.

Für den BSA-AKS +**0ALL**ALLxxxxxxx\xxxxxxxx=**VTV**xxxxxxxxxxxxxxxx**xxxxxxx**xxxxx ergibt sich somit die ID **0ALLVTV**. Eine redundante Anlage wird im AKS-Teil Anschluss gekennzeichnet. Für einen solchen BSA-AKS +**0ALL**ALLxxxxxxx\xxxxxxxx=**VTV**xxxxxxxxxxxxxxxx**02xxxxx**xxxxx ergibt sich die ID **0ALLVTV2**.

In den ALL_DPUA_BLE.xml und DPUA_*BSAID*.xml Files werden die Datenpunkte gruppiert abgelegt. Ein Datenpunkt kann dabei einer von 6 Gruppen zugeordnet sein. Diese Zuordnung basiert anhand der getroffenen Einstellungen im DxNode config des Datenpunktes. Tabelle 4 zeigt die Zuordnungsmatrix.

Tabelle 4: Gruppenzuordnung

Gruppe	Befehl	Meldung	Alarmquittierung	Beschreibung
*GRPRF*_WR	X			BLE schreibt, BSA liest
*GRPRF*_RD		X		BLE liest, BSA schreibt
*GRPRF*_RW	X	X		BLE und BSA schreiben und lesen
*GRPRF*_WRA	X		X	BLE schreibt, BSA liest Bidirektionale Alarmquittierung
*GRPRF*_RDA		X	X	BLE liest, BSA schreibt Bidirektionale Alarmquittierung
*GRPRF*_RWA	X	X	X	BLE und BSA schreiben und lesen Bidirektionale Alarmquittierung

Der Gruppenprefix GRPRF setzt sich aus den AKS-Teilen Streckenabschnitt, Anlage und Anschluss des AKS der BSA, getrennt mit einem `_`, zusammen.

Für den BSA-AKS +0**ALL**ALLxxxxxxx\xxxxxxxx=**VTV**xxxxxxxxxxxxxxxx**xxxxxxx**xxxxx ergibt sich somit der Gruppenprefix **ALL_VTV**. Bei einer redundanten Anlage mit dem BSA-AKS +0**ALL**ALLxxxxxxx\xxxxxxxx=**VTV**xxxxxxxxxxxxxxxx**02xxxxx**xxxxx ergibt sich der Gruppenprefix **ALL_VTV2**.

Abbildung 33 und Abbildung 34 zeigen ein Beispiel der generierten ALL_DPUA_BLE.xml und DPUA_0ALLVTV.xml Files. Zu erkennen ist, dass der lokale Name des Datenpunktes unterschiedlich ist (wie im ParaTool definiert, siehe Abbildung 32) aber den gleichen Netzwerknamen besitzt.

```
<DxUaCnfExt>
    <DPList>
        <Group gn="ALL_VTV_WR">
        <Group gn="ALL_VTV_RD">
        <Group gn="ALL_VTV_RW">
        <Group gn="ALL_VTV_WRA">
        <Group gn="ALL_VTV_RDA">
            <P a="A_VTVGEXI_ANLML1_STATU.cmdio.state_01" n="+Axxxxxxxxxxxxx\xxxxxxxx=VTVALLGExxxxANLML1x\xxxxxxxBEDAR" t="int32"/>
        </Group>
        <Group gn="ALL_VTV_RWA">
    </DPList>
```

Abbildung 33: Beispiel ALL_DPUA_BLE.xml

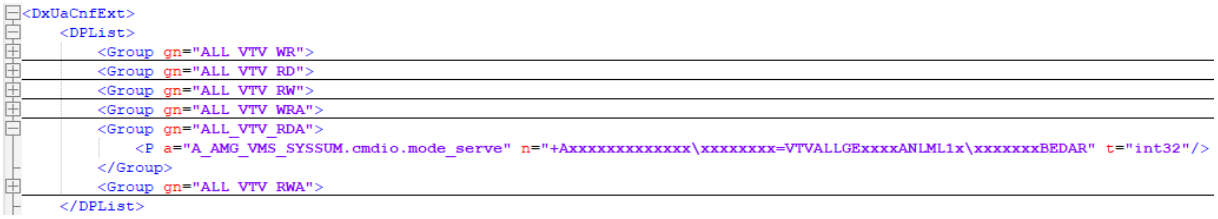


Abbildung 34: Beispiel DPUA_0ALLVTV.xml

Abbildung 35 zeigt ein Beispiel des generierten `ALL_MXUA_BLE_A.xml`. Die Verbindungsnamen (`cn`) setzen sich dabei aus den AKS-Teilen Streckenabschnitt, Anlage, Anschluss des AKS der BSA und der BLE Knoten Verbindung „**BLEA**“ oder „**BLEB**“ zusammen.

Für den BSA-AKS `+0`**`ALL`**`ALLxxxxxxx\xxxxxxxx=`**`VTV`**`xxxxxxxxxxxxxxxx\`**`xxxxxxx`**`xxxxx` ergibt sich somit der Verbindungsname **`ALL_CONN_VTVBLEA`**. Bei einer redundanten Anlage mit dem BSA-AKS `+0`**`ALL`**`ALLxxxxxxx\xxxxxxxx=`**`VTV`**`xxxxxxxxxxxxxxxx\`**`02xxxxx`**`xxxxx` ergibt sich der Verbindungsname **`ALL_CONN_VTV2BLEA`**.

```
<DxUaCnfExt>
    <ConnectionConfig>
        <Connect cn="ALL_CONN_VTVBLEA" type="DxUa" url="opc.tcp://127.20.164.55:48011">
            <VarCon>
                <P gn="ALL_VTV_WR" n="*" w="="/>
                <P gn="ALL_VTV_RD" n="*" r="="/>
                <P gn="ALL_VTV_RW" n="*" r="=" w="="/>
                <P gn="ALL_VTV_WRA" n="*" w="=" attr="A"/>
                <P gn="ALL_VTV_RDA" n="*" r="=" attr="A"/>
                <P gn="ALL_VTV_RWA" n="*" r="=" w="=" attr="A"/>
            </VarCon>
        </Connect>
```

Abbildung 35: Beispiel ALL_MXUA_BLE_A.xml

Das `CXUACL_`*`BSAID`*`.xml` File wird nur erzeugt, wenn der Knotentype der BSA `DxNode UA+Classic` ist, also Reflexe und Infotags mit anderen BSAs mittels DxClassic ausgetauscht werden müssen. In so einem Fall besitzt die BSA einen DxUA und einen DxClassic Knoten. Dieses File beinhaltet die aktive Verbindungsdefinition für den DxUA um Reflexe und Infotags mit dem DxClassic austauschen zu können. Die Lese/Schreibrichtung ergibt sich über die Einstellungen `Signal ist Senke` und `Signal ist Quelle` im `DxNode` config des Datenpunktes.

- `Signal ist Senke` Wert wird vom DxClassic gelesen
- `Signal ist Quelle` Wert wird zum DxClassic geschrieben

Für den Austausch wird dabei ausschliesslich der Netzwerkname (AKS) des Datenpunks verwendet. Abbildung 38 zeigt ein Beispiel eines generierten `CXUACL_0ALLVTV.xml`. Der Verbindungsname (`cn`) setzt sich dabei aus den AKS-Teilen Streckenabschnitt, Anlage, Anschluss des AKS der BSA und „**DXCL**“ zusammen.

Für den BSA-AKS `+0`**`ALL`**`ALLxxxxxxx\xxxxxxxx=`**`VTV`**`xxxxxxxxxxxxxxxx\`**`xxxxxxx`**`xxxxx` ergibt sich somit der Verbindungsname **`ALL_CONN_VTVDXCL`**. Bei einer redundanten Anlage mit dem BSA-AKS `+0`**`ALL`**`ALLxxxxxxx\xxxxxxxx=`**`VTV`**`xxxxxxxxxxxxxxxx\`**`02xxxxx`**`xxxxx` ergibt sich der Verbindungsname **`ALL_CONN_VTV2DXCL`**.

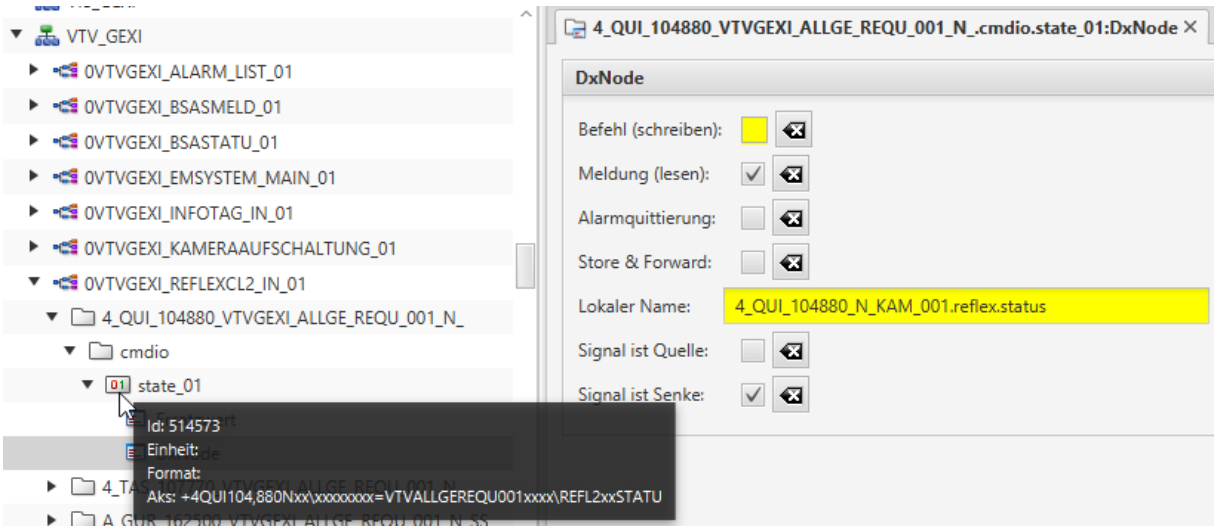


Abbildung 36: DxConfig Senkensignal

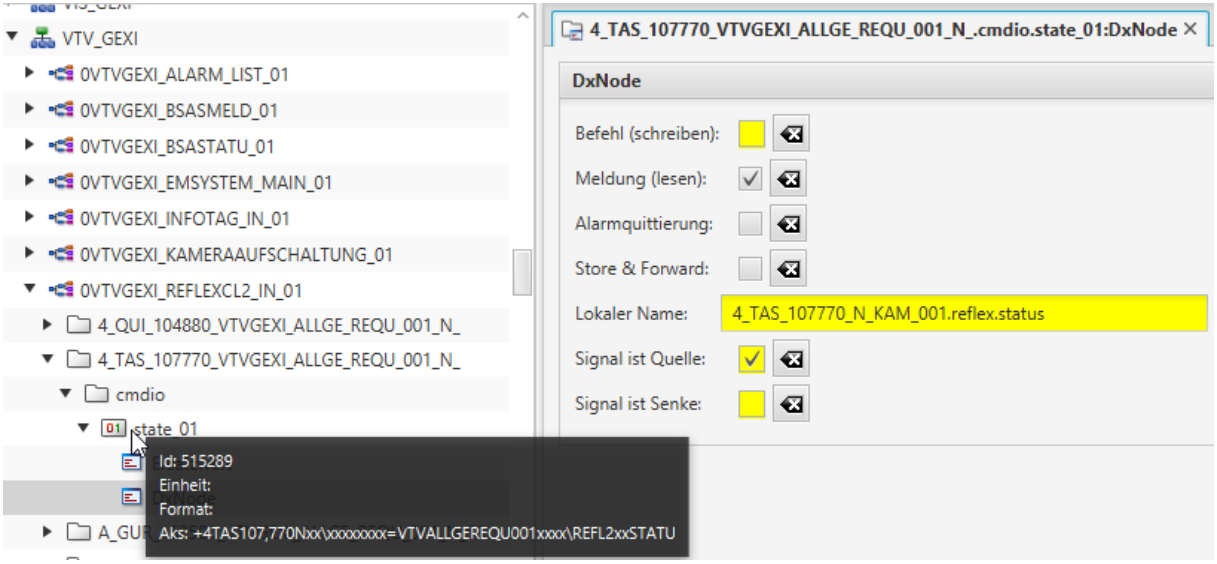


Abbildung 37: DxConfig Quellensignal

```
<DxUaCnfExt>
    <ConnectionConfig>
        <Connect cn="ALL_CONN_VTVDXCL" type="DxClassic" host="172.20.164.55" port="7580">
            <VarCon>
                <!--receive reflex/infotag-->
                <P n="+4QUI104,880Nxx\xxxxxxxx=VTVALLGEREQU001xxxx\REFL2xxSTATU" r="="/>
                <!--send reflex/infotag-->
                <P n="+4TAS107,770Nxx\xxxxxxxx=VTVALLGEREQU001xxxx\REFL2xxSTATU" w="="/>
            </VarCon>
        </Connect>
    </ConnectionConfig>
</DxUaCnfExt>
```

Abbildung 38: Beispiel CXUACL_0ALLVTV.xml

Kommunikationsmatrix

Die Kommunikationsmatrix definiert bei Anlagen mit Quellreflexen, welcher Quellreflex zu welchen Senkenreflexen anderer BSAs weitergeleitet werden soll. Bei der Aktivierung einer Kommunikationsmatrixversion einer BSA wird das File `CXUA_BSAID.xml` erstellt. Dieses File beinhaltet für jede involvierte Senkenanlage, die mittels DxNode OPC UA angebunden ist, eine aktive Verbindungsdefinition. Für den Austausch wird dabei ausschliesslich der Netzwerkname (AKS) der Datenpunkte verwendet. Die Verbindungsnamen (`cn`) setzen sich dabei aus den AKS-Teilen Abschnitt/Ort/Kilometer, Anlage und Anschluss des AKS der Quell-BSA und dem AKS-Teilen Anlage und Anschluss des AKS der Senken-Anlage zusammen.

Beispiel:

QuellBSA-AKS: `+GGSTBIERKELxxx\xxxxxxxx=BMAxxxxxxxxxxxxxxxx\xxxxxxxxxxx`
SenkenBSA-AKS: `+0ALLALLxxxxxxx\xxxxxxxx=VTVxxxxxxxxxxxxxxxx\xxxxxxxxxxx`
Verbindungsname: `BIERKEL_CONN_BMAVTV1.`

Beispiel redundante Senke:

QuellBSA-AKS: `+GGSTBIERKELxxx\xxxxxxxx=BMAxxxxxxxxxxxxxxxx\xxxxxxxxxxx`
SenkenBSA-AKS: `+0ALLALLxxxxxxx\xxxxxxxx=VTVxxxxxxxxxxxxxxxx\02xxxxxxxxx`
Verbindungsname: `BIERKEL_CONN_BMAVTV2.`

Beispiel redundante Quelle:

QuellBSA-AKS: `+GGSTBIERKELxxx\xxxxxxxx=BMAxxxxxxxxxxxxxxxx\02xxxxxxxxx`
SenkenBSA-AKS: `+0ALLALLxxxxxxx\xxxxxxxx=VTVxxxxxxxxxxxxxxxx\xxxxxxxxxxx`
Verbindungsname: `BIERKEL_CONN_BMA2VTV1.`

Beispiel Quelle und Senke redundant:

QuellBSA-AKS: `+GGSTBIERKELxxx\xxxxxxxx=BMAxxxxxxxxxxxxxxxx\02xxxxxxxxx`
SenkenBSA-AKS: `+0ALLALLxxxxxxx\xxxxxxxx=VTVxxxxxxxxxxxxxxxx\02xxxxxxxxx`
Verbindungsname: `BIERKEL_CONN_BMA2VTV2.`

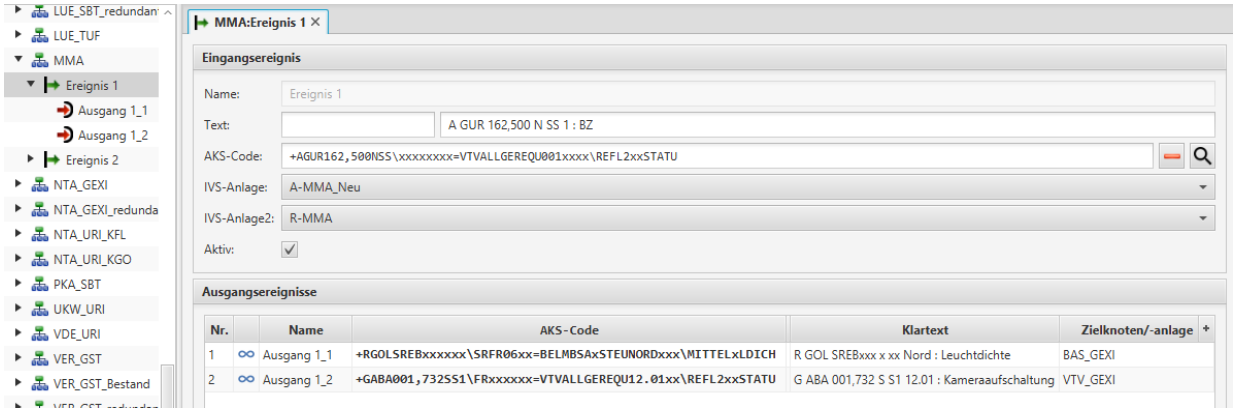


Abbildung 39: KommMatrix Konfiguration

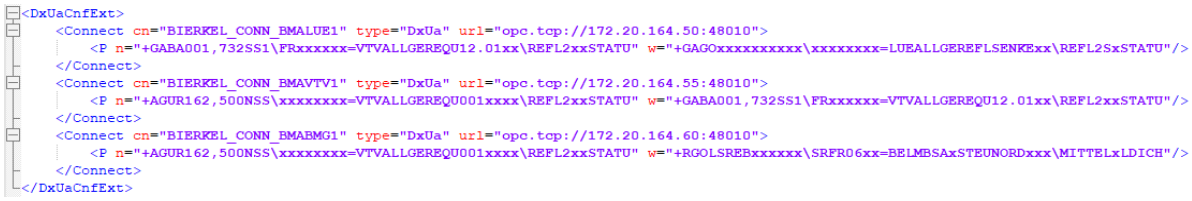


Abbildung 40: Beispiel CXUA_GGSTBLE.xml

11 Systemanforderung

Der DxNode OPC UA erfordert folgende Mindestvoraussetzungen für einen sicheren und stabilen Betrieb:

Betriebssystem

- Microsoft Windows 11 (64-Bit-Versionen)
- Microsoft Windows Server 2022 (64-Bit-Versionen)

Der Einsatz auf anderen Betriebssystemen oder älteren Windows-Versionen ist nicht zulässig.

Hardware

- Prozessor: Intel Core i5 mit 2,3 GHz oder vergleichbar
- Arbeitsspeicher (RAM): Mindestens 8 GB
- Festplattenspeicher: Mindestens 5 GB freier Speicherplatz
- Netzwerkadapter: 1 GBit/s Ethernet-Adapter

12 Installation DxNode OPC UA

Die notwendige Software für den DxNode OPC UA wird mit der «DxNodeUASetup.msi» installiert. Diese ist in der jeweils aktuellen Version für den UN BSA durch den zuständigen Projektleiter zu beziehen.

Hinweis: Benötigt die BSA-Anlage eine Verbindung, aufgrund Reflexe oder Infotags zu einer Anlage, welche den DxNode Classic in Verwendung hat, muss zusätzlich der DxNode Classic «DxNode.exe» installiert werden (siehe Kapitel 17).

Der Betrieb bzw. die Installation des DxNodeUA und DxClassic ist ausschliesslich auf einer Windows Umgebung möglich.

Schritt 1: Installation des Setups «DxNodeUASetup.msi»

Die Installation wird Standardmässig unter C:\DxNodeUA\ installiert. Alle DxNodeUA Knoten Verzeichnisse werden auf den BSA wie folgt angelegt:

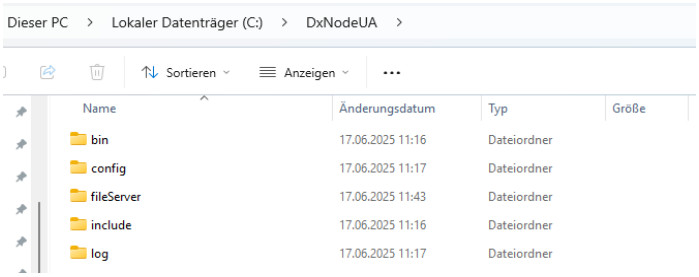


Abbildung 41: Ordnerstruktur DxNode OPC UA

Schritt 2: Konfiguration Hauptkonfigurationsfile

Das Hauptkonfigurationfile (C:\DxNodeUA\config) «`DxNodeUaConfig.xml`» welches in der Installation als Template mitgeliefert wird, muss Projektspezifisch durch den UN BSA umbenannt und konfiguriert (Siehe Kapitel 12.1) werden.

Der Filename bildet sich wie folgt:

- a - Erfassungsbereich (z.B. G für Gotthard)
- bbb - Streckenabschnitt (z.B. GST für Gotthardstrassentunnel)
- ccc - Abschnit/Ort/km (Optional, z.B. LRU für LZ_Ruetenen)
- ddd - Anlage (z.B. VER Verkehr)
- eeeeeeee - Fixer Postfix «DXNODEUA»

Beispiel: `G_GST_VER_DXNODEUA.xml`

Schritt 3: Installation als DxNodeUA Service

Folgender Befehl muss in der Kommandozeile ausgeführt werden.

```
c:\DxNodeUA\bin>
DxNodeUA.exe -install -startImmediate
             -conf c:\DxNodeUA\config\<Hauptkonfigurationsfile>.xml
```

Schritt 4: Start des DxNodeUA

Die Include – und Kommmatrixfiles werden anhand der ausgefüllten Integrationsformulare mittels ParaTool automatisch generiert und auf den Fileservern der BLE abgelegt.

Bei Start des DxNodeUA Knoten der BSA werden die Include- und Kommmatrix Files über den Fileserver heruntergeladen (Sofern eine Verbindung zum Fileserver vorhanden ist). Können keine Files heruntergeladen werden, wird der Knoten nicht gestartet. Alternativ können die Files manuell in die Verzeichnisse kopiert werden.

Schritt 5: Kontrolle ob konfigurierte Verbindungen verbunden sind

Über die Kommandline oder im Log-File kann geprüft werden, ob die konfigurierten Verbindungen zu den jeweiligen BSA bzw. BLE erfolgreich hergestellt wurde.

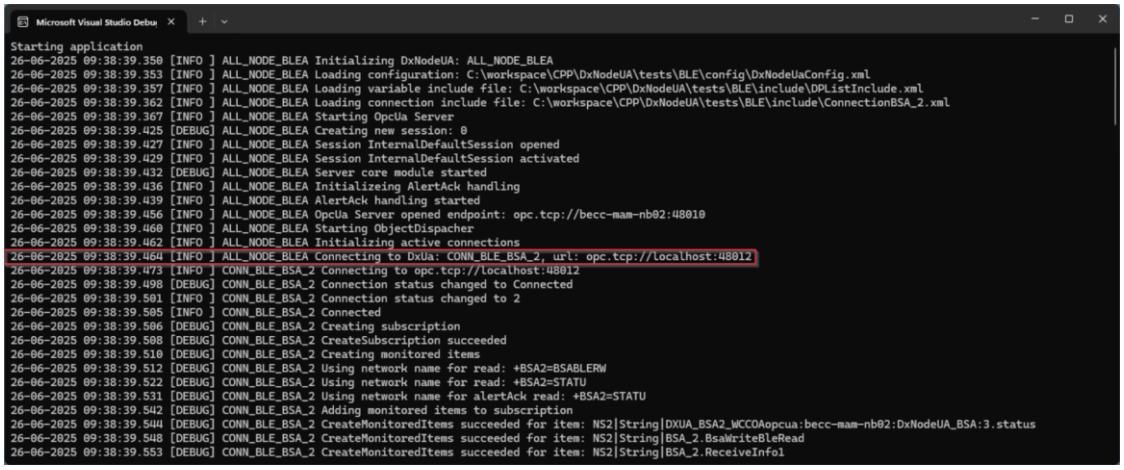


Abbildung 42: Auszug Log DxUA

12.1 Kommandline Optionen

Dem DxNode OPC UA können beim Starten mehrere Kommandlineoptionen mitgegeben werden (siehe Tabelle 5). Zwingend erforderlich ist die Option `-conf` um das Hauptkonfigurationsfile mit absolutem Filepath anzugeben, welches für den Hochstart benötigt wird:

```
DxNodeUA.exe -conf c:\DxNodeUA\config\DxNodeUaConfig.xml
```

Tabelle 5: Kommandline Optionen

Option	Beschreibung
`-conf`	Definiert das Hauptkonfigurationsfile.
`-install`	Weist den DxNode OPC UA an, sich als Windows Service (DxNodeUA) zu registrieren.
`-remove`	Weist den DxNode OPC UA an, sich aus der Windows Serviceliste zu entfernen. Hierbei ist die Angabe des Hauptkonfigurationsfiles mit `-conf` nicht erforderlich. Hinweis: Dienst muss gestoppt sein.
`-startImmediate`	Optional, startet den DxNode OPC UA nach der Installation als Windows Service
`-h`	Zeigt eine Hilfe zu allen Kommandline Optionen an.

Beispiel normales Starten in der Konsole:

```
c:\DxNodeUA\bin>DxNodeUA.exe -conf c:\DxNodeUA\config\DxNodeUaConfig.xml
```

Beispiel Installieren als Service:

```
c:\DxNodeUA\bin>DxNodeUA.exe -install -conf c:\DxNodeUA\config\DxNodeUaConfig.xml
```

Beispiel Entfernen aus der Serviceliste:

```
c:\DxNodeUA\bin>DxNodeUA.exe -remove
```

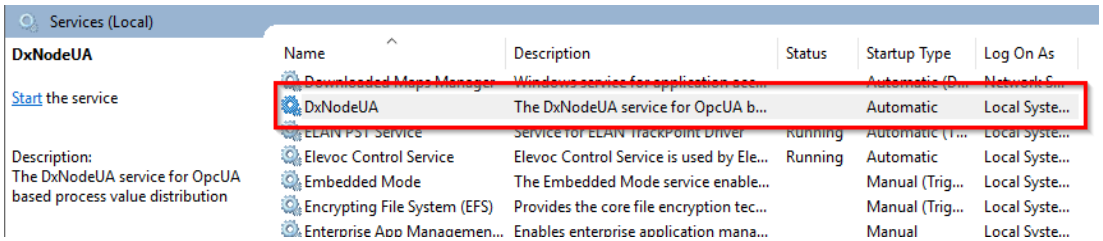


Abbildung 43: Beispiel DxNodeUA als Service installiert

13 Anbindungsschnittstelle für BSA

Die BSA-Anlagen kommunizieren ausschließlich über die OPC UA DA (Data Access)-Schnittstelle mit dem DxNode OPC UA. Der DxNode OPC UA stellt dabei direkt am jeweiligen Knoten einen OPC UA Server zur Verfügung, über den die Datenübertragung erfolgt. Es ist ausdrücklich vorgeschrieben, dass keine anderen Schnittstellen oder Kommunikationsprotokolle für die Anbindung der BSA-Anlagen zulässig sind. Alle Verbindungen müssen über diese standardisierte OPC UA DA-Schnittstelle realisiert werden, um eine einheitliche und sichere Kommunikation sicherzustellen. Um eine Alarmquittierung zu ermöglichen, erfolgt die Übertragung über einen eigenen Datenpunkt (siehe Kapitel 5.4).

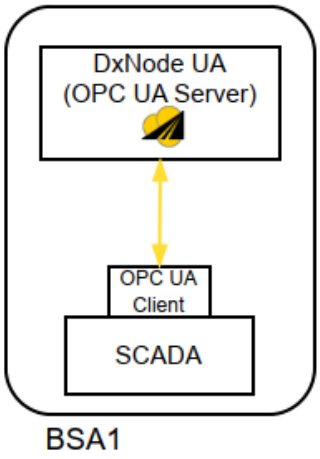


Abbildung 44: BSA Anbindung OPC UA

14 Anwendungsbeispiel SCADA-Verbindung

Um sich mit dem DxNode OPC UA zu verbinden, können die SCADA Systeme ihre jeweiligen nativen OPC UA Clients einsetzen. Im Folgenden werden Konfigurationsbeispiele anhand von WinCC OA gezeigt.

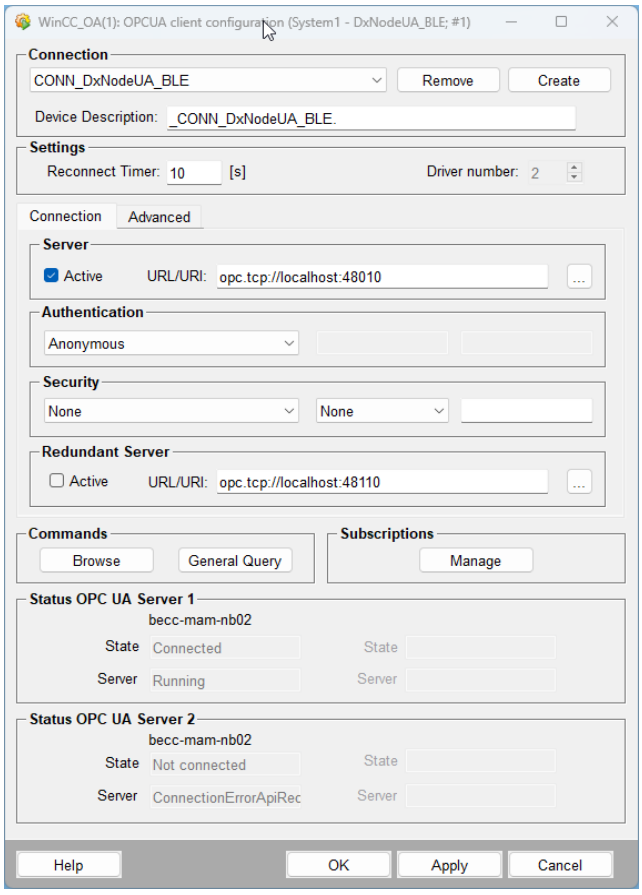


Abbildung 45: WinCC OA OPC UA Driver Konfiguration

```
[opcua]
timestampDefaultForRead = 1              #uses timestamp from server
timestampWriteMode = 3    #sets server and source timestamp to _stime
```

Abbildung 46: WinCC OA zusätzliche config Einträge

14.1 Variable Lesen

Um Werte einer Variable vom DxNode OPC UA zu lesen, muss eine entsprechende Subscription und die NodeId der Variable definiert werden. Die Direction muss auf Input gesetzt sein.

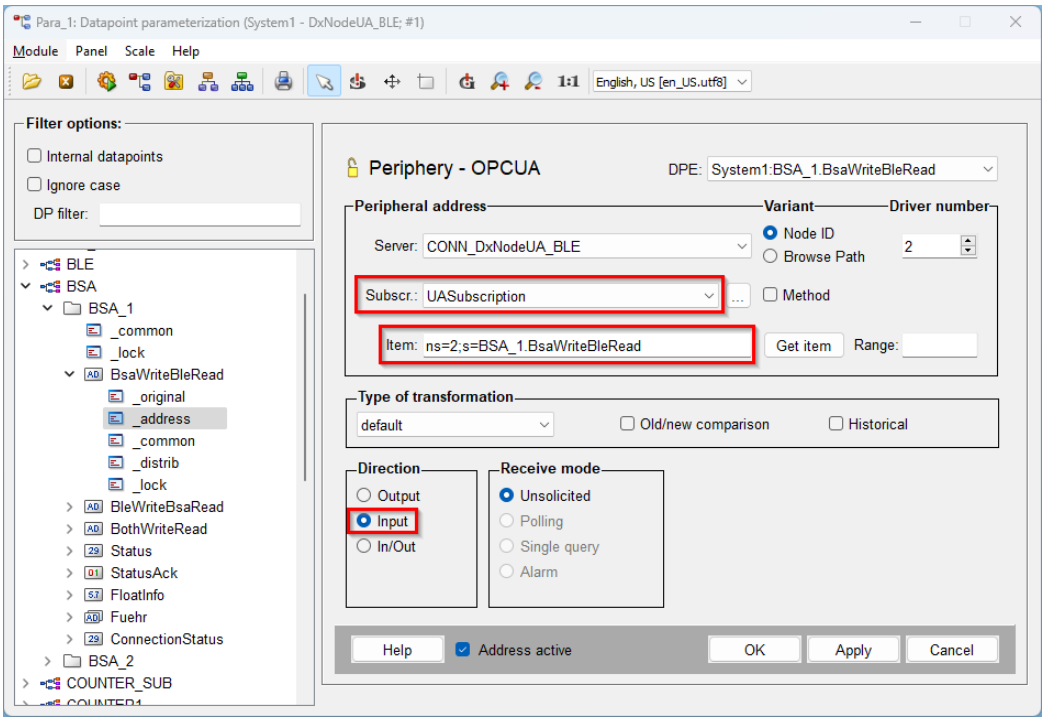


Abbildung 47: Beispiel Variable Lesen

14.2 Variable Schreiben

Um Werte in eine Variable des DxNode OPC UA zu schreiben, muss die NodeId der Variable definiert werden. Die Direction muss auf Output gesetzt sein.

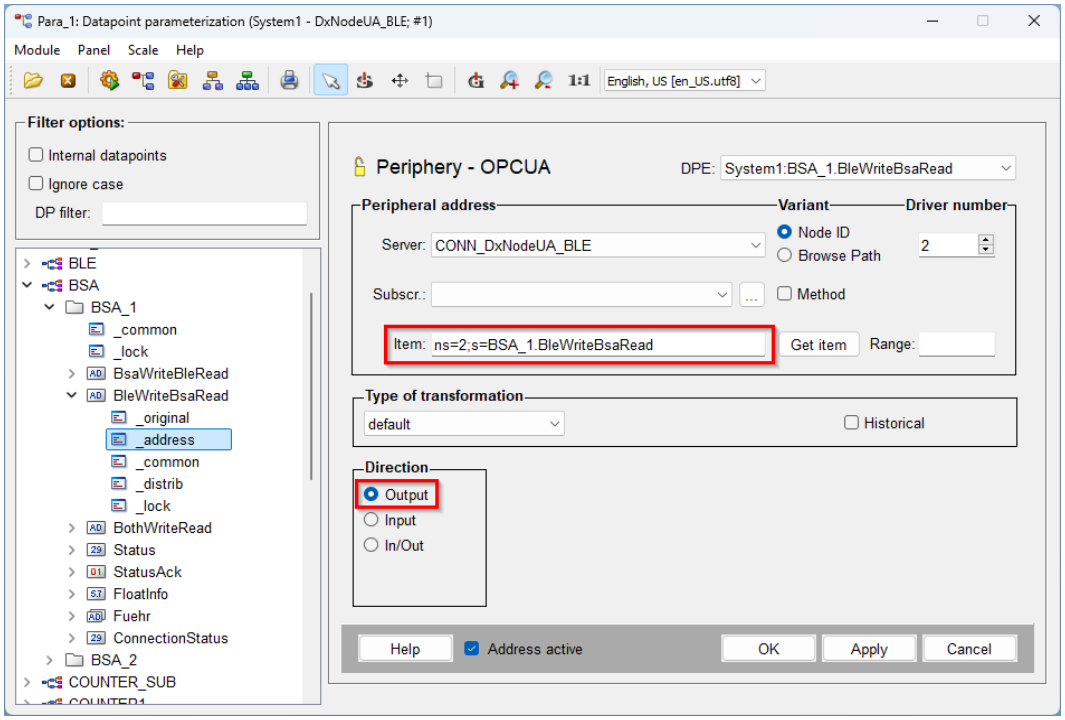


Abbildung 48: Beispiel Variable Schreiben

14.3 Variable Lesen und Schreiben

Um Werte einer Variable vom DxNode OPC UA sowohl zu lesen als auch in diese zu schreiben, muss eine entsprechende Subscription und die NodeId der Variable definiert werden. Die Direction muss auf In/Out gesetzt sein.

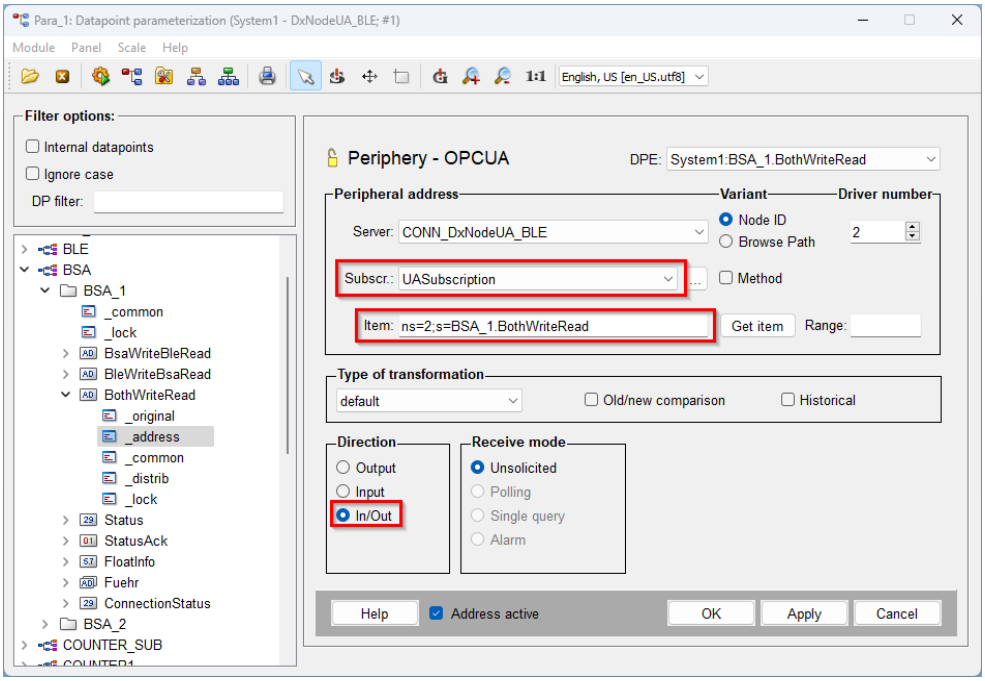


Abbildung 49: Beispiel Variable Lesen und Schreiben

14.4 Variable mit Alarmquittierung

Für jede Variable für die Quittier Information transportiert wird, muss im SCADA System ein zusätzliches boolsches Datenpunkelement angelegt werden und diese mit der jeweiligen `alertAck` Variable lesend und schreibend verbunden werden. Dieses Datenpunkelement kann dann für die weiteren SCADA Mechanismen bezüglich Alarmquittierung verwendet werden.

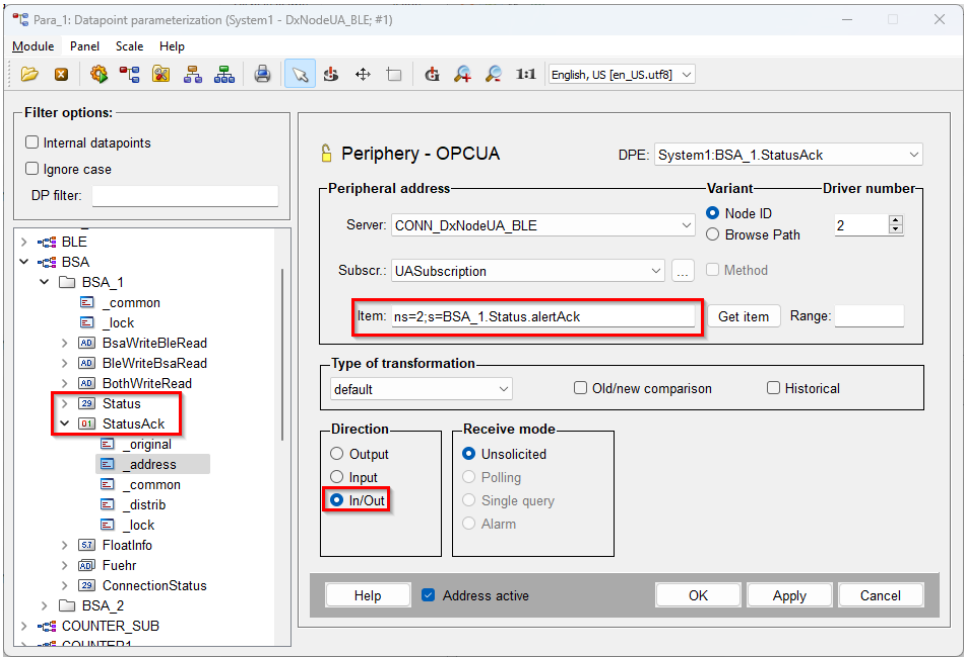


Abbildung 50: Beispiel AlertAck

15 Darstellung des DxNode OPC UA Knotens im BLE und BSA HMI

15.1 Knotensymbol

Das DxNode OPC UA Symbol wird immer im Zusammenhang mit einem Rechnersymbol dargestellt und zeigt an, dass auf dem betreffenden Abschnittsrechner ein DxNode OPC UA Knoten läuft.

Beispiel:

Zusätzlich werden die Alarme des Knoten in der Meldezeile sowie am Objekt gemäss TS07 dargestellt.

Hinweis: Hat die BSA-Anlage einen DxNode Classic und ist Reflextechnisch eine Quellenanlage, sind diese Verbindungsalarme auch anzuzeigen.

16 Versionshistorie DxNode OPC UA

Versionshistorie:	
DxNodeUA Version 0.1	Erstimplementierung

17 Zusatzinformationen DxNode Classic Knoten

Die Beschreibung dient dazu, um bei Erfordernis den DxNode Classic zu installieren.

Checkliste für die Installation/den Betrieb des DxNode Classic (ALT: TINWUR) Knotens

Die Software (DxNode.exe) in der jeweils aktuellen Version ist für den UN BSA bei der zuständigen Projekt- bzw. Bauleitung zu beziehen.

Der bevorzugte Betrieb des DxNode Knotens im Rahmen der BLE/N2 ist der Betrieb als **Windows Service**.

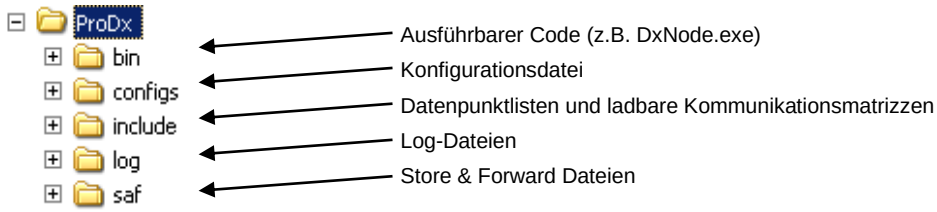
1. Schritt:

Anlegen eines Projektverzeichnisses für den DxNode Knoten auf den BSA, entsprechend folgender Vorgaben und Struktur:

- Laufwerk: C:\
- Verzeichnis: ProDx
- Verzeichnis freigegeben: NEIN

Alle DxNode Knoten Verzeichnisse sollen auf den BSA wie folgt angelegt sein:

C:\ProDx



2. Schritt:

Anlegen des BSA spezifischen XML Config:

Hierfür wurde ein „DxNode Template“ erstellt, welches dem UN BSA die eigenständige Erstellung des XML Config erleichtert.

TINWUR Template: TinwurTemplateReal_20102009.xml

Jeder UN BSA muss dieses DxNode Template entsprechend folgender Vorgaben entsprechend anpassen:

a - Erfassungsbereich z.B. S fuer Seelisberg

bbb - Streckenabschnitt z.B. ARU fuer Ruetenen

ccc - Abschnit/Ort/km z.b. LRU fuer LZ_Ruetenen

ddd - Anlage z.B. BEL Beleuchtung, LUE Lueftung, FKA Funkanlage, usw

Beschreibungen:

Eingerichtete DxNode Verbindungen:

- lokale Anbindung im Realsystem

Verbindung zwischen TWK der BSA und Applikation der BSA

- BLE-Verbindungen im Realsystem

Echtzeit-Verbindung zwischen TWK der BSA und TWK der BLE

- Reflexverbindungen im Realsystem

Echtzeit-Verbindung zwischen TWK der BSA und TWK fremd BSA

Include- und Kommmatrixfile Download:

- Parametrierverbindung für zukünftigen Download im Realsystem

Echtzeit-Verbindung für den automatischen Download der Includefiles und der Kommmatrixfiles

Hinweis:

Der Dateiname der XML Configdatei ist nicht frei wählbar, sondern ergibt sich aus den AKS Codes der BSA, z.B:

L_TUF_BEL.xml

Dabei gilt:

L → Erfassungsbereich L

TUF → Streckenabschnitt (z.B. Tunnel Flüelen)

BEL → Anlage (z.B. Beleuchtung)

3. Schritt:

Start des DxNode -Knotens als Batch Datei :

Starten einer Batch-Datei mit folgendem Inhalt, oder Eingabe der Befehle in einem Command-Window:

DxNode -app -projdir <Projektverzeichnis> <kompletter Pfad von XML Config>

Optionale Parameter:

-snd, -rcv → Zeigen gesendete und empfangene Telegramme in der Dos Box des Knoten an.

Beispiel:

```
c:
cd\
cd C:\ProDx\bin
C:\ProDx\bin\DxNode.exe –app -projdir C:\ProDx C:\ProDx\configs\{Name des XML Config}.xml
```

4. Schritt:

Anlegen der BSA spezifischen Include- und Kommmatrixfiles:

Die Include – und Kommmatrixfiles werden anhand der ausgefüllten Integrationsformulare mittels ParaTool automatisch generiert und auf den Fileservern (host=172.22.88.44, 172.22.76.44) der BLE abgelegt.

Sobald der DxNode Knoten der BSA in der BLE Domain das erste Mal gestartet wird, erfolgt der automatische Include- und Kommmatrixfiles download. Die durch das ParaTool erzeugten Files werden in dem angelegten „include“ – Ordner des ProDx Verzeichnisses abgelegt.

Wenn die BSA noch nicht in der BLE Domain integriert ist, kann kein automatischer Download erfolgen und die notwendigen Files müssen manuell „include“ – Ordner des ProDx Verzeichnisses kopiert werden.

Hinweis:

Den Import der Integrationsformulare in das ParaTool übernimmt der UN BLE.

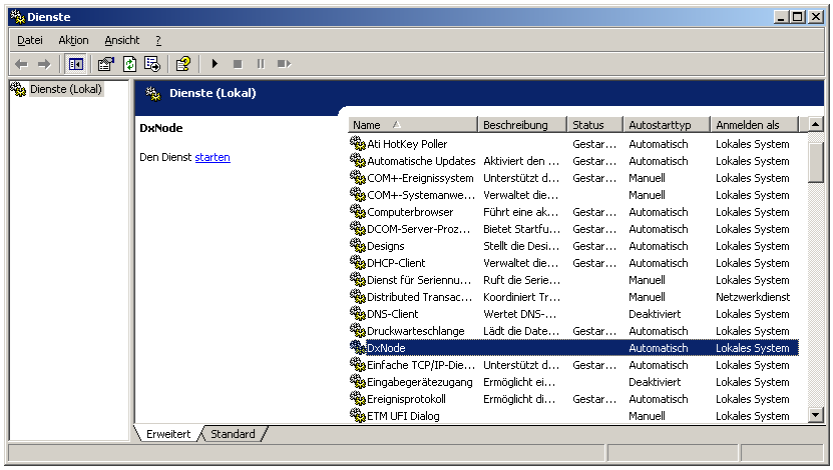
5. Schritt:

Installation und Start des DxNode -Knotens als Service:
DxNode -install -projdir <Projektverzeichnis> <kompletter Pfad von XML Config>

Beispiel:

c:
cd\
cd C:\ProDx\bin
C:\ProDx\bin\DxNode.exe –install -projdir C:\ProDx C:\ProDx\configs\{Name des XML Config}.xml

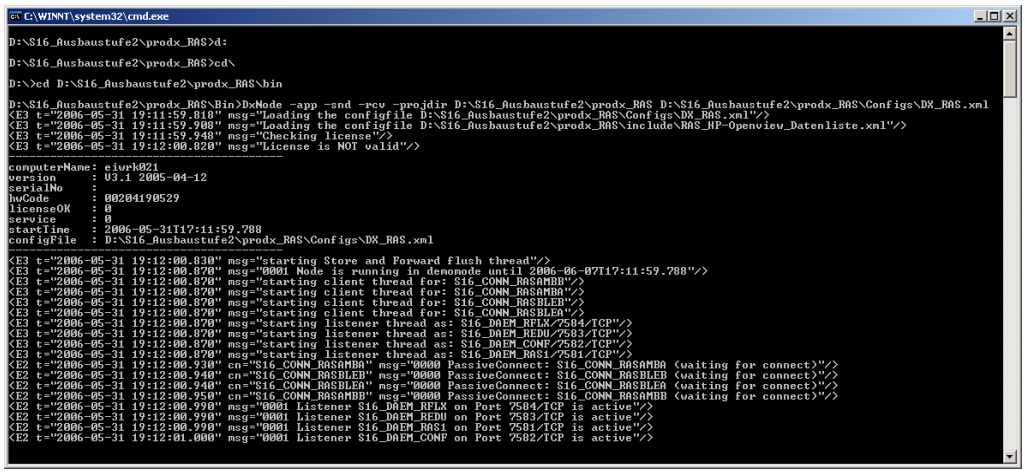
Serviceeinstellungen vornehmen (unter : Systemsteuerung – Verwaltung – Dienste)



Den Dienst als „automatisch“ starten konfigurieren und danach beim ersten Mal über obiges Fenster starten. Danach sollte der Dienst (DxNode) automatisch bei jedem Rechnerneustart gestartet werden.

6. Schritt:

Kontrolle der Dos Box eines gestarteten DxNode -Knotens ob die eingerichteten Verbindungen tatsächlich verbunden sind:



Beispiel:

localhost:

<E2 t="2009-09-10 08:54:27.060" msg="accepted connection for KFL_DAEM_WVA1 from IP 127.0.0.1 (0)"/>

BLE A1:

<E2 t="2009-09-10 08:54:28.091" msg="accepted connection for KFL_DAEM_WVA1 from IP 172.22.88.34 (1)"/>

BLE B1:

<E2 t="2009-09-10 08:54:31.216" msg="accepted connection for KFL_DAEM_WVA1 from IP 172.22.76.34 (2)"/>