DxWare.com

DxWare AG
Postfach 332
CH-6373 Ennetbürgen

Phone +41(0)41 620 02 90
Mail dxware@DxWare.com
Web www.DxWare.com

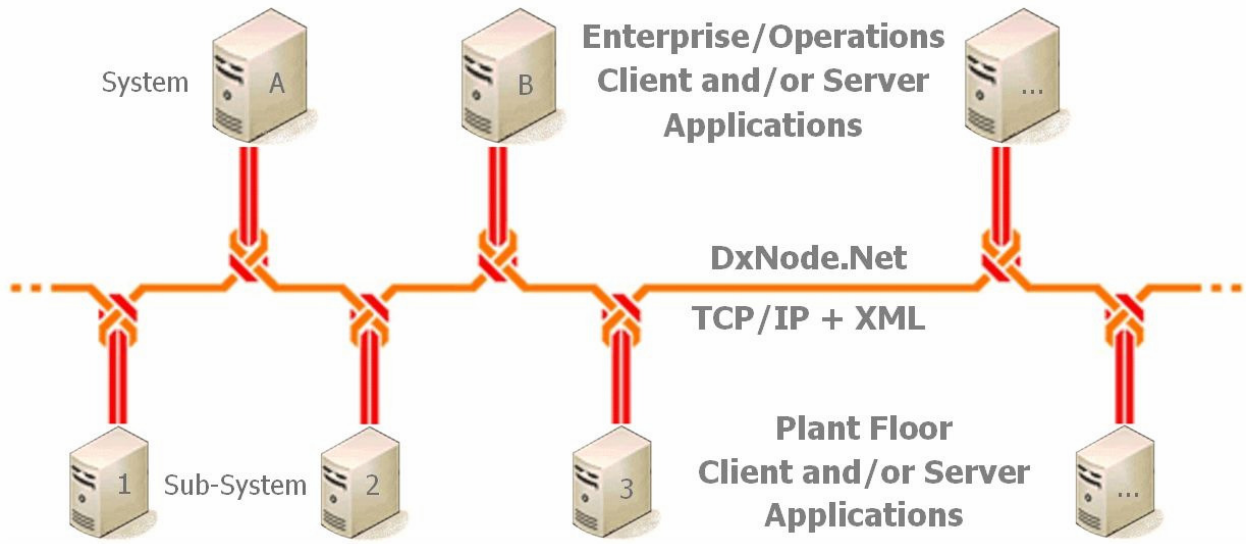
DxNode®Net

Data e**x**change **Node Net**work

Netzwerk für Datenaustausch

Referenz Manual

Ausgabe 2.15



Inhalt

Einführung 4
Was ist DxNode.Net ? 4
Anforderungen für DxNode.Net 4
Merkmale und Vorteile 5
DxNode.Net ist überall dort einsetzbar wo 5
Übersicht 6
Zielgruppe 6
Netzwerk für Datenaustausch 6
Konzepte und Funktionen 8
Client/Server Verhalten 8
Parametrierung statt Programmierung 9
Verbindungsaufbau und -ablauf 10
Verbindungs- und Daten-Monitoring 11
Transaktionen 13
Zeitstempel und Quality 13
Dateninhalte und Datentypen 14
Signalkonditionierung 14
Store & Forward Meldungen 14
Redundante Anbindungen 15
XML Standard 15
Eigennamen 15
Wildcards 15
Konfiguration 16
Darstellungskonventionen 17
XML Elemente 17
XML Attribute 17
Lokale Konfiguration <DxCnfLoc> 18
Grundeinstellungen <Node> 18
Zugangsport <Daemon> 18
Datenpunktliste <DPList> 19
Anschluss/Verbindung <Connect> 20
Kommunikationsmatrix <Matrix> 25
Externe Konfiguration <DxCnfExt> 26
Externe Datenpunktliste <DPList> 26
Externe Anschluss/Verbindung <Connect> 26
Externe Kommunikationsmatrix <Matrix> 27
Beispiele für Lokale und Externe Konfiguration 28
Nur Lokale Konfiguration 28
Lokale Konfiguration und Externe Referenz 29
Lokale Konfiguration und Externe Referenz mit Download 30
Proprietäre Konditionierungsdaten <C> 31
Attribut Definition und Verwendung 32
a="Address (application)" n="Name (network)" 32
alive="Alive Timeout" 33
attr="Attributes" 34
c="Event Counter" (reserved) 35
cn="Connect Name" 35
cn_list="Connect List" (DxMonitor) 36
config_level="Configuration Level" 37
config_version="Configuration Version" 37
dn="Daemon Name" 38
encoding="XML Encoding" 38
f="Format and Datatype" siehe auch v=".." 39
fast_update="Control Flag" siehe upd_delay=".." 40
flush_cycle="Store&Fwd Flush Cycle" 40
fn="File Name" 40
from="Source File" to="Target File" 42

gn="Group Name" 43
host="Host Name or IP-Address" 44
i="Index" siehe auch v=".." 45
ip="IP Address" (DxMonitor) 46
ip_list="IP Address List" (DxMonitor) 46
l="Length of Message" (reserved) 46
max_conn="Max Number of Connects" 47
max_telegr_length="Max Length of Telegram" (reserved) 47
msg="Clear Text Message" 47
n="Name (network)" siehe a=".." 48
nid="Node ID" sid="Sender ID" rid="Receiver ID" 48
nn="Node Name" 49
path="Working Directory" 50
perf_cycle="Performance Check Interval" 50
port="Port Addr or Service-Nr" 51
prefix="Internal DP Name Prefix" 52
pwd="Password" 52
q="Quality" siehe auch v=".." 53
r="Read from Server" w="Write to Server" 54
rd_interval="Read Interval" wr_interval="Write Interval" 56
reconnect_cycle="Reconnect Cycle" 57
rid="Receiver ID" siehe nid=".." 57
s="Status and Bit Filter" siehe auch v=".." 57
sid="Sender ID" siehe nid=".." 58
store_fwd_buffer="Store&Fwd Buffer" 58
t="Timestamp" siehe auch v=".." 58
tc="Timestamp Corrective" siehe auch v=".." 59
tgt="Time Greater Than" tlt="Time Less Than" 59
tlt="Time Less Than" siehe tgt=".." 60
to="Target File" siehe from=".." 61
tt="Time Tolerance" 61
u="Engineering Unit" siehe auch v=".." 61
upd_delay="Update Delay" fast_update="Control Flag" 62
v="Value" siehe auch t=".." q=".." f=".." s=".." i=".." u=".." x=".." c=".." 64
verbose="Trace Flags" (DxMonitor) 66
version="XML Version" 67
w="Write to Server" siehe r=".." 67
wr_interval="Write Interval" siehe rd_interval=".." 68
x="Text Information" siehe auch v=".." 68
Anhang **69**
OPC Unified Architecture und DxNode.Net 69
DxNode Client/Server Verbindungen 71
DxNode Interne Datenpunkte 74
Bezeichnungskonvention 74
Datenpunkte Allgemein <Node> 74
Datenpunkte pro Zugangsport <Daemon> 75
Datenpunkte pro Anschluss/Verbindung <Connect> 75
Knotenstatus und Verbindungszustände 76
DxNode Telegrammstrukturen und XML Schema 77
Konfigurationsbeispiele 78
DxNode Installation als Applikation oder Service 81
DxNode Lizenzierung 81
Internetadressen für XML-Editoren und Tools 82
DxNode Einbindungsworkshop 82

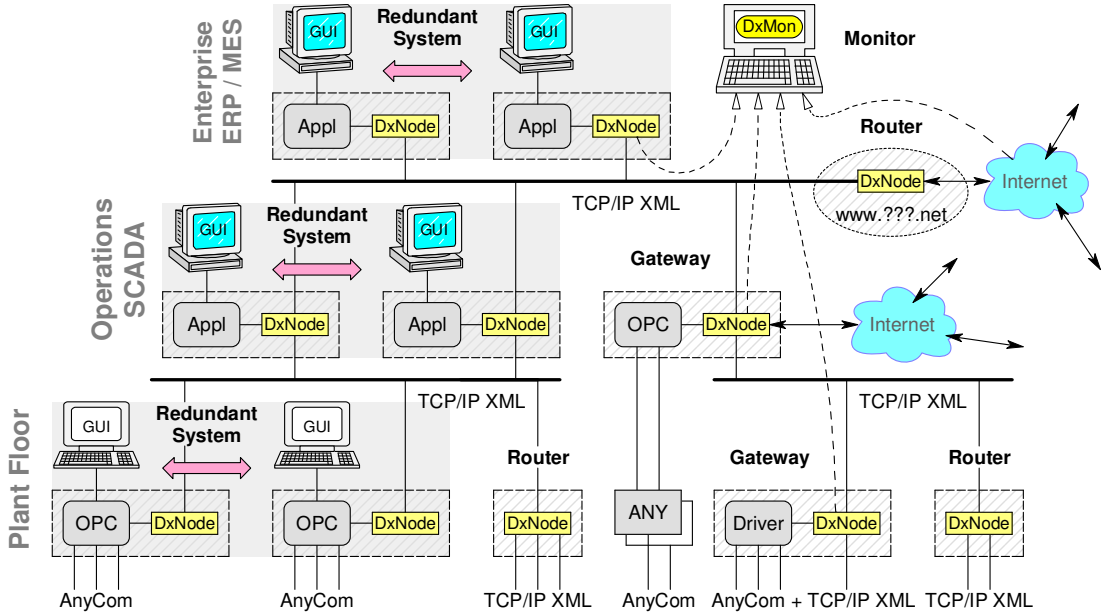
Änderungsgeschichte

Version	Datum	Status	Bearbeiter	Änderungsgeschichte
2.0	2006-08-24	Release	F. Weber	Zweite Version komplett überarbeitet
2.10-2.14	2007-08-22	Release	F. Weber	Textkorrekturen, Beispiele / Installation / Lizenzierung
2.15	2008-03-05	Release	F. Weber	Ergänzungen De-Installation / Textkorrekturen

Einführung

Was ist DxNode.Net ?

- **DxNode.Net** ist eine komplette, lauffähige Software die als so genannte Knoten (**Node**) in den beteiligten Systemen installiert wird. Die einzelnen Knoten sind beliebig vernetzbar - auch über Internet - und bilden zusammen das Netzwerk (**Net**) für den Datenaustausch (**D**ata e**X**change) zwischen den Systemen
- DxNode.Net ist eine offene, neutrale Kommunikationsschicht (Middleware), die den Anwender unabhängig von Herstellervorgaben macht. Die Software ist für Linux und Microsoft Windows Plattformen verfügbar und kann bei Bedarf als C++ Source Code hinterlegt werden.
- Die Ankopplung wie auch der Datenaustausch zwischen DxNode erfolgt über ein einziges Protokoll, das auf den internationalen Standards **TCP/IP** (Transport), **XML** (Datendarstellung) und **Web-Services** (Internet) basiert und mit **XSD** (XML Schema) dokumentiert ist. Die Konfiguration ist ebenfalls XML basiert und kann mit dem selben XML Schema geprüft werden
- DxNode.Net unterstützt **OPC** Client und Server Schnittstellen sowie weitere nützliche Funktionen wie Speichern und Weiterleiten von Daten (Store&Forward), mehrfach redundante Ankopplungen, Datenverschlüsselung (Encryption) u.a.m.
- DxNode.Net trennt Kommunikation und Anlagen d.h. die Systeme können sich nicht gegenseitig beeinträchtigen. Ähnlich wie bei der Post braucht sich der Anwender nicht um den sicheren Transport der Daten zu kümmern, das - wie auch andere Funktionen z.B. "Einschreiben" oder "Rückantwortaufforderung" - übernimmt DxNode.Net
- DxNode.Net hat sich in grossen Projekten bewährt. Das Netzwerk bildet eine Art verteilte Datenbank, jede Anlage kann mit jeder anderen Daten austauschen, auch über OPC und Internet:



Anforderungen für DxNode.Net

- Das DxNode.exe Programm bzw. der Dienst (Service) ist verfügbar für Microsoft Windows und Linux
- Die Standards TCP/IP, XML und Web-Services werden praktisch von allen Systemen unterstützt
- Das DxNode.exe Programm benötigt mit allen Libraries ca. 500 kByte plus ca. 200 Byte pro Datenpunkt, abhängig von den Bezeichnungen und Daten
- DxNode kann über Internet kommunizieren wenn die entsprechenden Ressourcen wie Internet Anschluss (z.B. mit DSL) und Web-Services verfügbar sind

Merkmale und Vorteile

- Plattformunabhängiges Netzwerk mit installierbaren, komplett lauffähigen Knoten (Node) für den Datenaustausch (Data eXchange) zwischen beliebigen Systemen
- Instandhaltbarkeit von grossen/komplexen Anlagen (Investitionsschutz) durch klare Trennung von Kommunikation und Systemen
- Offene, neutrale Kommunikationsschicht (Middleware) mit integrierten Funktionen Speichern und Weiterleiten (Store&Forward), Redundanz, Datenverschlüsselung usw.
- Anwendbar für alle Bereiche - Prozess, SCADA, MES, ERP mit hoher Verfügbarkeit auch im Internet
- Weltweite Übertragung von beliebigen Echtzeitdaten und OPC Daten im Internet
- Bedienbarkeit mit Transitionsanzeige von beliebigen Stellen auch via Internet
- Übersicht und Transparenz durch zwei Adressräume für Netzwerk und Anlage
- Kunden-Standard für Datenpunkt Namenskonventionen ist mit XML Schema an anpassbar
- Einfache XML Ankopplung mit XSD Validierung (XML Schema Compliance Test)
- Installierbares Programm/Dienst mit XML Konfiguration → mit XML Schema validierbar
- Einfache Ankopplung mit XML Protokoll → Telegramme mit XML Schema validierbar
- Schnittstelle basiert auf TCP/IP (Transport), XML (Datendarstellung) und Web-Services (Internet)
- Ereignis gesteuert für alle Transaktionen, Data Access und Events (kein Datenverlust)
- Automatische Zeitstempelung in Millisekunden für alle Daten
- Unterstützt 100'000 Datenpunkte pro DxNode mit Durchsatz > 1'000 DP/sec
- Signalbeobachtung/Aufzeichnung mit DxMonitor für Unterhalt im Internet
- Datenpunktauswahl einzeln, in Gruppen und/oder mit komplexen Wildcards "*"
- Zentrale Parametrierung mit Download oder automatische Datenpunktgenerierung parametrierbar
- Ausbaubares/aufwärtskompatibles Design mit XML → unterstützt alle Sprachen/Zeichen
- Anpassungen sind schnell im Netzwerk wirksam, da nur ein Softwareprogramm beteiligt ist
- Ausgezeichnete Wartbarkeit, da Komponenten problemlos zu- und abgeschaltet werden können
- DxNode.Net verfügt über OPC Client und Server Schnittstellen und unterstützt alle wesentlichen Merkmale, die mit der OPC UA Spezifikation (Unified Architecture) geplant sind. DxNode.Net ist aber nicht nur eine Spezifikation, sondern eine lauffähige Software, die fertig installiert und konfiguriert werden kann z.B. für: ◆ Store&Forward (Transport ohne Datenverlust) ◆ Transitionsanzeige für Befehle (Bedienkomfort) ◆ Zwei Namensräume für Datenpunktbezeichnung (Lokal und Netzweit) ◆ Übertragung und Bedienung im Internet ◆ Verbindungsherstellung durch beide Partner, Client oder Server mit Verbindungsüberwachung ◆ Client und Server sind parametrierbare Funktionen (Roles) ◆ Beliebige Server-Server und Client/Server-Verbindungen ◆ Kopplung mehrfach redundanter Systeme mit automatischer Umschaltung+Datensynchronisation ◆ Einheitliche Funktionsebene und einheitliche Konfiguration ◆ Verschlüsselung der Daten usw.

DxNode.Net ist überall dort einsetzbar wo ...

- Eine grösstmögliche Unabhängigkeit von Herstellern und Lieferanten gefordert ist d.h. wenn lange Lebenszyklen für eine Anlage geplant sind
- Die Investitionen durch den Ersatz von Teilsystemen geschützt werden können
- Echtzeitdaten zwischen unterschiedlichen/heterogenen Systemen, System-Versionen oder Systemen verschiedener Hersteller ausgetauscht werden
- Echtzeitdaten über Internet ausgetauscht oder gesammelt anderen Systemen und Anwendungen zur Verfügung gestellt werden
- Eine komfortable Bedienung von Sollwerten und Eingaben auch in Systemen oder Netzen mit langsamer Reaktionszeit (z.B. Internet) gefordert ist
- Ein hierarchischer Aufbau der Netzwerke oder eine sichere Trennung der Zugriffe für Firmennetz (Enterprise), Bedienung (Operations) und Prozess (Plant Floor) gefordert ist
- Redundante Systeme gekoppelt werden müssen, Verbindungen können mehrfach redundant und mit Store&Forward ausgelegt werden, auch über Internet oder für OPC
- OPC Daten über mehrere Systeme, Netzwerke oder über Internet ausgetauscht werden

Übersicht

Zielgruppe

Dieses Handbuch ist für Anwender, Ingenieure und Techniker gedacht, die sich im Bereiche der Automation und Kommunikation auskennen. Die Ausführungen sollen dem Anwender das System näher bringen und ihn befähigen, die Funktionen zu verstehen um ein Netzwerk parametrieren zu können.

Das gesamte Netzwerk wird incl. der Prozess-Anbindungen mittels **XML** parametriert. XML steht für "e**X**tensible **M**ark-up **L**anguage", ein weltweiter Standard als Beschreibungssprache für Dokumente und Daten von der Grundform: <**X a**=".." **b**=".." **c**=".."/>, wobei **X**=Element und **a,b,c**=Attribute mit den Werten ".." darstellen. Ein Anwender braucht also keine Programmierkenntnisse, ein Grundwissen über XML ist jedoch sinnvoll, entsprechende XML-Tools und Editoren unterstützen ihn dabei.

Externe Anbindungen sind als **TCP**/**IP** Schnittstellen zu programmieren, wobei die Parametrierung für die Anbindung in XML zur Verfügung gestellt wird. Ein Programmierer braucht - neben der Kenntnis über die anlagenseitige Ankopplung - im Wesentlichen zu wissen, wie man eine TCP/IP Verbindung zum lokalen Zugangsport des Netzwerkes herstellt und wie man den XML-Datenstrom interpretiert bzw. übermittelt und empfängt. Dies kann in einem Workshop von DxWare AG vermittelt werden.

Zeichen und Darstellungen

- **Verweise auf andere Kapitel oder Dokumente** sind mit einem **Pfeilsymbol** ➲ markiert
- **XML Elemente und Attribute** sind in der XML-Notation dargestellt z.B. <**Connect**>, **nn**=".."
- **Erläuterungen zur Konfiguration** sind im Kapitel ➲ Darstellungskonventionen zu finden

Netzwerk für Datenaustausch

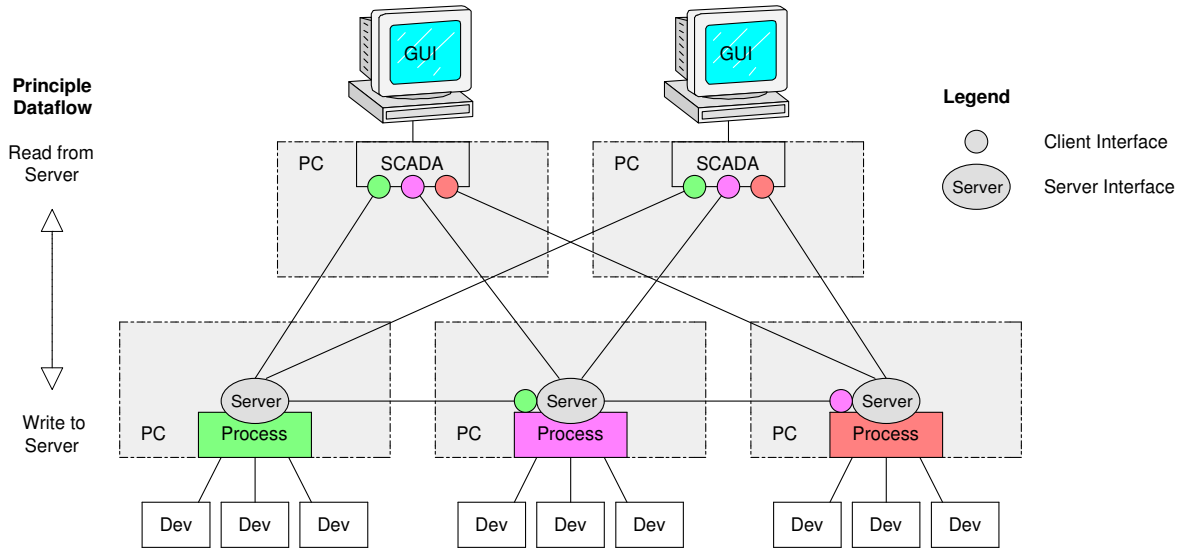
Das Netzwerk basiert auf dem Software-Knoten DxNode, ein standardisiertes Programm, das in allen beteiligten Rechnern installiert wird und die wesentlichen Funktionen im netzweiten Datenverkehr übernimmt. Alle DxNode zusammen bilden ein Netzwerk zum Datenaustausch (Data eXchange), auch Middleware (Software-Ebene) genannt, mit folgenden Merkmalen:

- Automatischer Datenabgleich zwischen den DxNode (Synchronisation)
- Verteilen sowie Speichern und Weiterleiten (Store&Fwd) von Daten und Meldungen
- Verlustfreie Übertragung von Ereignissen dank Pufferung für jeden Datenpunkt
- Beliebig hierarchische Netzwerkstrukturen incl. Server zu Server Verbindungen
- Ankopplung von redundanten Systemen auf allen Hierarchieebenen
- Konfiguration für den Datenaustausch im Netzwerk incl. aller anzubindenden Systeme
- Zwei gleichwertige Adressräume für Netzwerk (Standard) und Anbindungen (Proprietär)
- OPC kompatible Datendarstellung mit Value, Timestamp und Quality
- Erweiterte Datendarstellung mit Format, Status, Index, Engineering Unit und Klartext
- Transitionsanzeige beim Bedienen für alle Befehle (bi-direktional)
- Meldungstransport mit Unquittiert-Anzeige und Quittierung
- DxMonitor zum Bedienen/Beobachten und Prüfen aller Transaktionen von und nach DxNode
- Parametrierbarkeit aller Verbindungen incl. der proprietären Anbindungen in XML
- Parametrierbarer Zugangsport ermöglicht Zugriffsschutz auf TCP/IP-Ebene
- Passwort geschützte Verbindungen und Zugangsports
- Plattformunabhängig, basiert auf den weltweiten Standards XML, TCP/IP und C++
- Zukunftsgerichtete Erweiterbarkeit und Wartbarkeit des gesamten Systems
- Offene, einfache Anbindung basierend auf XML und TCP/IP oder OPC

Jeder Knoten (DxNode) verwendet dasselbe Programm (exe), das mit einer Konfigurationsdatei (xml) parametriert wird. DxNode kommuniziert mittels eines standardisierten XML-Protokolls über TCP/IP. Externe Anbindungen verwenden exakt die gleiche Schnittstelle, d.h. sie spielen aus der Sicht des Netzwerkes die Rolle eines normalen DxNode.

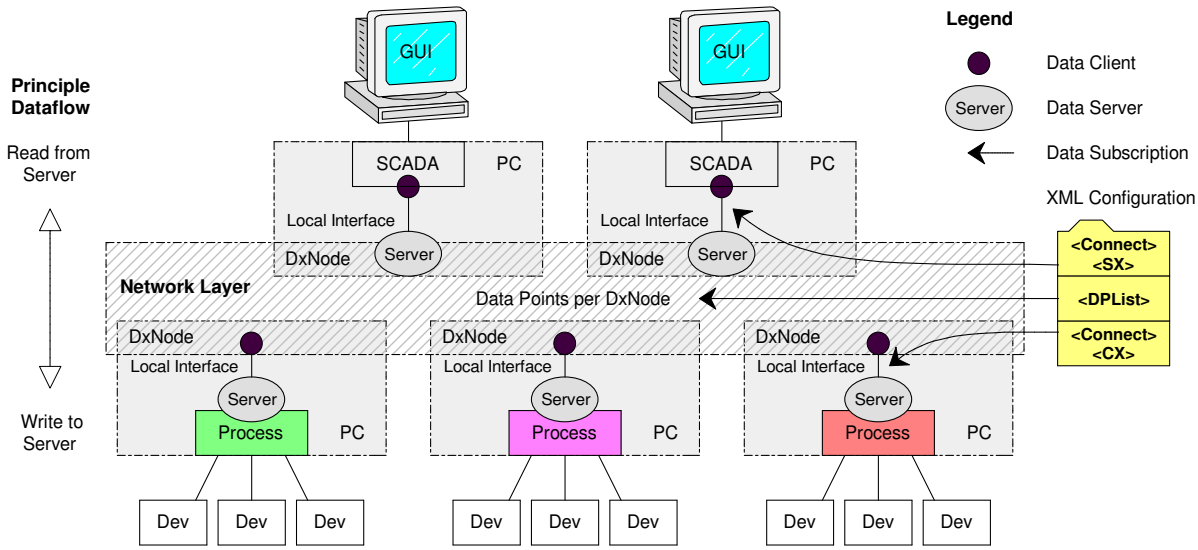
Die Parametrierung für das gesamte Netzwerk kann von einer zentralen Stelle aus erfolgen, dabei kann die Konfigurationsdatei von DxNode heruntergeladen werden (Download). Das Konzept erlaubt den Ausbau zu einem offenen, standardisierten Netzwerk mit einer nicht eingeschränkten Anzahl proprietärer Client und/oder als Server Anbindungen.

Jeder Anwender kommuniziert in DxNode.Net nur mit einem Teilnehmer, nämlich dem lokal installierten DxNode. Im Vergleich dazu müsste im **klassischen Client/Server-Modell** jeder Teilnehmer direkt mit jedem anderen kommunizieren. Dies hat in einem komplexen System entsprechende Auswirkungen:



- Jedes System muss mehrere, ggf. individuelle Anbindungen incl. Parametrierung unterstützen
- Redundante - und Server⇔Server Kopplungen sind nicht einfach realisierbar
- Netzwerkweite Funktionen und Standards sind nur schwierig zu realisieren und zu warten

DxNode.Net Middleware bietet - ähnlich einem Layer im OSI-Modell - Vorteile, indem wesentliche Funktionen des netzweiten Datenaustausches übernommen und standardisiert werden:



- Nur eine Client/Server Anbindung muss pro System parametriert und hergestellt werden
- Redundante - und Server⇔Server Kopplungen sowie andere Funktionen werden unterstützt
- Die Parametrierung für das gesamte Netzwerk incl. Anbindungen ist mit XML standardisiert

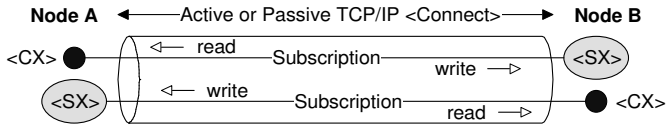
Jeder DxNode besitzt ein **Prozessabbild** mit den lokal relevanten Datenpunkten. Diese werden in einer **Datenpunktliste** <**DPList**> definiert und für lokale bzw. netzweite **Verbindungen** <**Connect**> mit einer so genannten **Client oder Server Kommunikationsmatrix** <**CX**> bzw. <**SX**> ausgewählt um sie im Partnerknoten für den Versand bzw. Empfang anzumelden (subskribieren). Die Parametrierung für das gesamte Netzwerk kann von einer zentralen Stelle aus erfolgen, wobei die Konfigurationsdatei von DxNode heruntergeladen werden kann (Download).

Konzepte und Funktionen

Client/Server Verhalten

Im klassischen Konzept ist der **Client**=**Dienstanforderer** und der **Server**=**Diensterbringer**. Dies gilt für die Herstellung der Verbindung (der Client verbindet sich aktiv mit dem Server) wie auch für die angeforderte Dienstleistung (der Client fordert Informationen vom Server an). Allerdings ist mit diesem Konzept keine beliebige Vernetzung realisierbar, weil ein Teilnehmer entweder Client oder Server ist.

DxNode kann sowohl die Rolle des Clients wie auch die eines Servers spielen. Zudem kann DxNode eine Verbindung zu einem anderen Knoten herstellen, unabhängig davon, ob er die Rolle des Clients oder Servers spielt. Innerhalb des Netzwerkes unterscheiden wir daher zwischen **aktivem und passivem Anschluss** <**Connect**> und verwenden die Begriffe Client/Server ausschliesslich für die Rolle, die ein Teilnehmer in Bezug auf die auszutauschenden Daten spielt, d.h. **Client**=**Datenanwender** <**CX**> und **Server**=**Datenquelle** <**SX**>. Die Anmeldung (Subskription) wird mit der Verbindungsherstellung erteilt. Dabei hat die Anmeldung als Client automatisch eine Server Subskription im Partnersystem zur Folge und umgekehrt. Eine Subskription <**CX**> und/oder <**SX**> ist - unabhängig ob aktiv oder passiv - nur auf einer Teilnehmerseite zu definieren, d.h. jede Verbindung kann, unabhängig ob aktiv oder passiv, gleichzeitig die Rolle eines Clients wie auch die eines Servers unterstützen:



DxNode kann die gleichen Daten z.B. über eine erste Verbindung als Client erhalten und sie über andere Verbindungen als Server zur Verfügung stellen. Dies ermöglicht die Vernetzung von Knoten über mehrere Hierarchiestufen (Chaining). Dabei ist zu beachten, dass Daten grundsätzlich nur eine Quelle haben (in redundanten Systemen sind es zwei oder mehrere), sich jedoch über beliebige Netzwerk-Verbindungen baumförmig verteilen können. Daraus ergeben sich Datenrichtungen, die wir **Read from Server** (Server→Client) und **Write to Server** (Client→Server) nennen. Sowohl Client wie auch Server können Daten in beide Richtungen d.h. **bi-direktional** (Read/Write) austauschen. Dies ist nur deshalb möglich, weil sich Client und Server in Bezug auf die Übermittlung von Daten unterschiedlich verhalten:

- Ein Client sendet einen vom Server erhaltenen Wert nie an diesen zurück d.h. der Wert wird nur dann zum Server geschrieben, wenn er tatsächlich im Client - oder ursprünglich von einem anderen Client verändert wurde. Andererseits sendet ein Server einen veränderten Wert immer an alle Clients (incl. den verursachenden) und zwar unabhängig davon wer den Wert ursprünglich verändert hatte.
- Der Datenabgleich (Synchronisation) beim Hochfahren des Netzwerkes bzw. von DxNode findet ausschliesslich vom Server zum Client und nie in umgekehrter Richtung statt, d.h. alle Daten werden zur Synchronisation von der Quelle gelesen (Read from Server) und ggf. überschrieben, wenn sie während des Abgleichvorganges verändert (Write to Server) wurden.

Diese Festlegungen sind für eine Parametrierung des Netzwerkes wichtig, im Wesentlichen muss man aber nur darauf achten, dass die Daten im Netzwerk genau eine Quelle haben. Die Quelle ist der Server (in redundanten Systemen sind es mehrere), der den Wert ursprünglich erzeugt bzw. besitzt. Für das besprochene Client/Server Verhalten gelten zusammengefasst folgende **Regeln**:

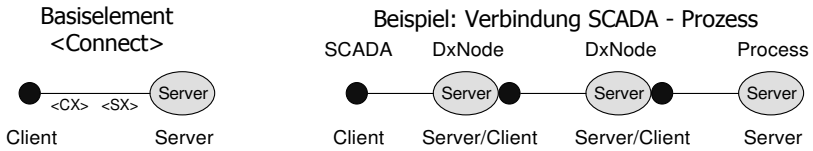
- Transaktionen <**SX**>→<**CX**> werden **Read** (lesen) genannt, <**CX**>→<**SX**> **Write** (schreiben)
- Daten werden in Leserichtung <**SX**>→<**CX**> synchronisiert, Schreiben ist dann blockiert
- Besitzer der Daten ist <**SX**>, alle Änderungen werden immer an alle <**CX**> weitergeleitet
- Benutzer der Daten ist <**CX**>, nur lokale Änderungen werden an <**SX**> geschrieben
- Für eine Verbindung <**Connect**> können eine <**CX**> und/oder eine <**SX**> subskribiert werden
- Nur ein DxNode wird für <**CX**>/<**SX**> parametriert, der Partner übernimmt die komplementäre Rolle

Parametrierung statt Programmierung

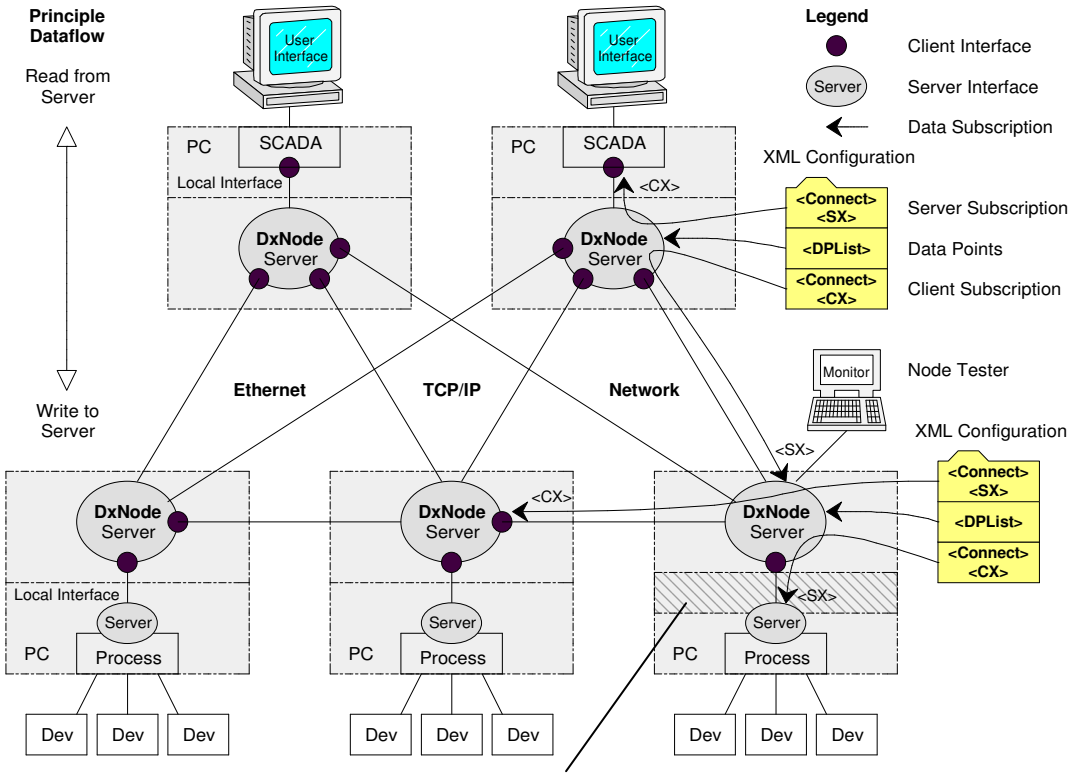
DxNode.Net ist komplett parametrierbar. Dies gilt auch für allfällig zu programmierende proprietäre Anbindungen. Die Konfigurationsdatei besteht aus wenigen Elementen, die wichtigsten davon sind:

- die **Datenpunktliste** <**DPList**> definiert alle Datenpunkte (DP) eines DxNode gekennzeichnet mit dem **n**="Namen" des netzwerkweiten Kennzeichnungssystems sowie mit der **a**="Adresse" der lokal anzubindenden Anlage und/oder weiteren Attributen z.B. zur Gruppierung der Datenpunkte. DxNode verfügt also über zwei Adressräume **n**=".." und **a**=".." die gleichwertig genutzt werden können und damit die lokale Anbindung vereinfachen.
- die **Kommunikationsmatrix** <**Matrix**> ist als Datenpunktliste für den Datenaustausch mit anderen Partnern zu verstehen. Dabei wird pro **Verbindung** <**Connect**> eine Auswahl von Datenpunkten festgelegt, um diese mit dem Verbindungsaufbau beim Partner anzumelden (Subscription). Partner können die angekoppelte Anlage oder jeder andere DxNode sein. Mit der Subskription wird festgelegt, ob der lokale DxNode der Datenanwender d.h. **Client** <**CX**> oder die Datenquelle d.h. **Server** <**SX**> ist ➲ Client/Server Verhalten. Für jede Verbindung kann eine <**CX**> und/oder eine <**SX**> angemeldet werden.

Im Prinzip ist das System ein Netzwerk von beliebig parametrierbaren Client/Server Verbindungen. Dabei stellen die Anlagen nichts anderes als Knoten dar, die zwar eine andere Funktion als DxNode haben, aber für die Anbindung genauso konfiguriert werden. Tatsächlich kann man das gesamte Netzwerk durch zusammenfügen von Basiselementen vom Typ <**Connect**> darstellen:



Das Basiselement <**Connect**> ist als konfigurierbare Schnittstelle zu verstehen. Die Datenverteilung und deren Parametrierung werden mit DxNode.Net zur Verfügung gestellt. Weil DxNode über zwei Adressräume verfügt, muss der Anwender primär seine eigenen Adressen kennen - er muss sich nicht um Bezeichnungen oder um den netzweiten Datenaustausch kümmern. In der Grafik sind die Konfigurations-Dateien mit den relevanten Elementen <**DPList**> und <**Connect**> (gelb) hervorgehoben:



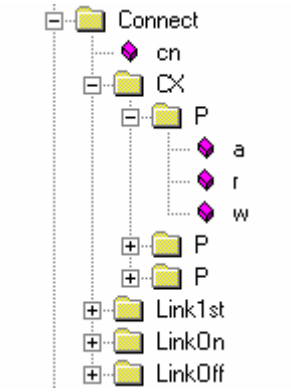
Auch proprietäre Anbindungen (schraffiert) sind mit DxNode.Net konfigurierbar.

Verbindungsaufbau und -ablauf

Der gesamte Datenaustausch basiert auf XML Telegrammen <**X0**>, die Datenpunkte <**P**> aus der Datenpunktliste enthalten. Weitere XML Elemente der Konfiguration werden in den Telegrammen <**X0**> verwendet z.B. dient die Matrix <**CX**> oder <**SX**> zur Subskription von Datenpunkten.

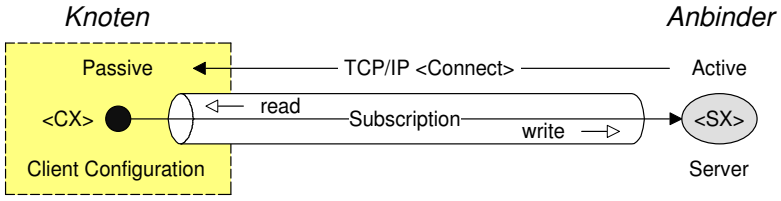
Beispiel: Verbindungsablauf und XML Telegramme für eine Server-Anbindung mit Client Matrix <**CX**> im Knoten (eine Client-Anbindung mit Server Matrix <**SX**> im Knoten funktioniert sinngemäss):

Konfiguration



Aufgabenstellung

Ein externer **Anbinder** soll sich **aktiv** als **Datenserver** an eine passive Verbindung <**Connect**> mit der Client Matrix <**CX**> an DxNode ankoppeln.



Element <**CX**> enthält eine Auswahl von Datenpunkten <**P**> mit den lokalen Adressen **a**=".." des Anbinders, die angemeldet (subskribiert) und übertragen werden sollen. Die Datenpunkte werden im lokalen DxNode als Client genutzt und gehören dem Anbinder als Server (owner).

Schritt	Verbindungsablauf mit XML Telegrammen
Beginn	**DxNode** startet und erstellt Defaultwerte gemäss Instruktionen in Element <**Link1st**>
1. ←	**Anbinder** erstellt TCP/IP Verbindung zu Rechner Name **host**=".." mit <**Daemon**> **port**=".."
2. →	**DxNode** bestätigt TCP/IP Verbindung mit Socket Handle und startet Daemon-Prozess
3. ←	**Anbinder** fordert die Vermittlung mit dem vorkonfigurierten <**Connect**> **cn**=".." an ... <**X0** [**t**=".."][**sid**=".."][**rid**=".."]><**Connect cn**=".."><**Switch**/></**Connect**></**X0**>
4. →	**DxNode** bestätigt die angeforderte Vermittlung mit Telegramm ... <**X0** [**t**=".."][**sid**=".."][**rid**=".."]><**ConnectR cn**=".." /></**X0**> dann subskribiert DxNode die lokale <**CX**> gemäss Konfiguration, führt die Initialisierung gemäss Instruktionen in Element <**LinkOn**> durch und übermittelt die korrespondierende <**SX**> ... <**X0** [**t**=".."][**sid**=".."][**rid**=".."]> <**SX**> <**P a**=".." **r**=".." [**w**=".."]/> <**P a**=".." **r**=".." [**w**=".."]/> <**P a**=".." **r**=".." [**w**=".."]/> </**SX**> </**X0**> ☞ Der Anbinder erhält automatisch die für ihn geltende komplementäre Subskriptions-Liste <**SX**> mit dem exakt gleichen Inhalt wie die parametrierte <**CX**>
5. ←	**Anbinder** übermittelt als Antwort auf <**SX**> alle Datenpunkte zur Initialisierung mit <**SXR**> ... <**X0** [**t**=".."][**sid**=".."][**rid**=".."]> <**SXR**> <**P a**="."><**D v**=".." [**t**=".."][**q**=".."]/></**P**> <**P a**="."><**D v**=".." [**t**=".."][**q**=".."]/></**P**> <**P a**="."><**D v**=".." [**t**=".."][**q**=".."]/></**P**> </**SXR**> </**X0**> ☞ <**SXR**> kann auch weggelassen werden, es bedeutet Anfang <**SXR**> und Ende </**SXR**> der Initialisierung. Daten <**D**> können auch als Event <**E**> übermittelt werden.
6. →	**DxNode** sendet Befehls-Datenpunkte der <**CX**> mit **Write** Daten <**e**..> (advise) ... <**X0** [**t**=".."][**sid**=".."][**rid**=".."]><**P a**="."><**e v**=".." [**t**=".."][**q**=".."]/></**X0**>
7. ←	**Anbinder** sendet geänderte Datenpunkte der <**SX**> mit **Read** Daten <**E**..> (update) ... <**X0** [**t**=".."][**sid**=".."][**rid**=".."]> <**P a**="."><**E v**=".." [**t**=".."][**q**=".."]/></**P**> <**P a**="."><**E v**=".." [**t**=".."][**q**=".."]/></**P**> </**X0**>
Abbruch	**DxNode** erstellt Werte gemäss Instruktionen in Element <**LinkOff**> bei Verbindungsabbruch. (Verbindungsabbau z.B. mit <**Connect**> Telegramm, Task Terminierung oder Alive Verlust).

Legende: Attributwerte ".." sind Daten in ASCII, Attribute in eckigen Klammern [..] sind optional

Die **Schritte 1..4** können mit einem C++ Programmaufruf **con.NewClient**("**cn**", "**host**", "**port**") der DxNode Library (NodeAPI) ausgeführt werden. Wenn die Verbindung **cn**=".." steht, ist die Variable **con.sock** > **0** und ein **Alive** Signal wird automatisch zwischen DxNode und Anbindungsprogramm ausgetauscht. Die Variable kann zur Freigabe von Prozeduren zum Datenaustausch Verwendet werden.

Die **Schritte 5..7** kann der Anbinder weitgehend bestimmen, er muss nicht zwingend eine Response <**SXR**> absetzen, wenn er alle Daten mit <**P**> Telegrammen übermittelt. Der Read und Write Datenaustausch geschieht ab Schritt 6 spontan bei Änderungen und völlig asynchron.

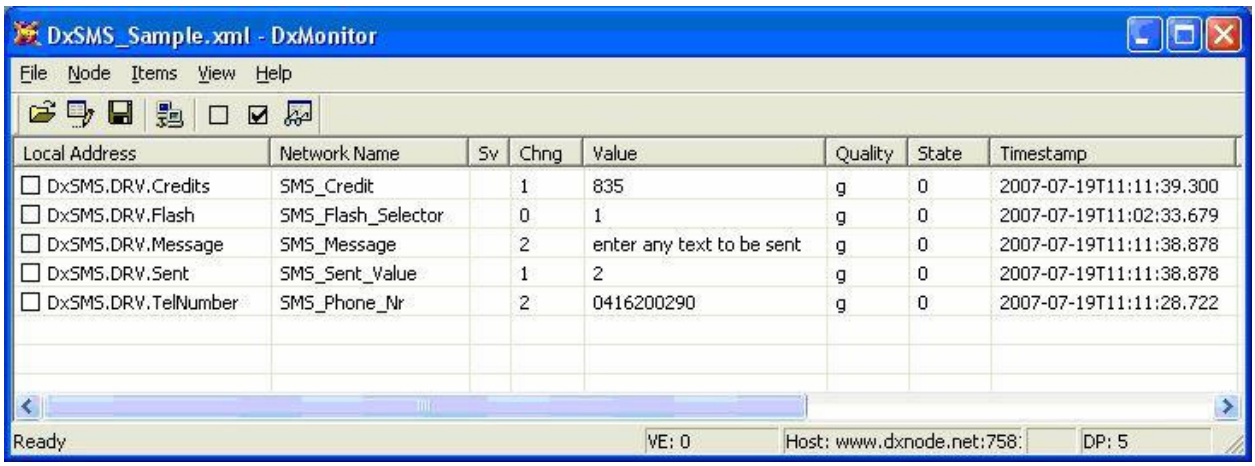
Die **allgemeine Form der XML Telegramme auf TCP/IP Ebene** (ASCII, Stream-Socket) ist ...

hhhhhhhh<**Telegram** [**Attribute**=".."]>[<**Function** [**Attribute**=".."]]/>][<**Command**/>]</**Telegram**>

Dabei stellt "hhhhhhhh" einen Header als fixe 8-stellige Hexzahl die Länge (Anzahl ASCII-Zeichen) des vollständigen XML Telegrammes dar (Länge von <Telegram> bis incl. </Telegram>). Die Länge ist bei Anbindungen zu berücksichtigen, die ohne DxNode Library (NodeAPI) arbeiten.

Verbindungs- und Daten-Monitoring

DxNode wird mit einem Monitor geliefert, mit dem sich ein Benutzer passwortgeschützt auf beliebige, zugängliche DxNode einloggen kann. DxMonitor präsentiert die Datenpunkte der DxNode Datenbank oder einen Ausschnitt davon wie folgt:



Mit DxMonitor lassen sich die XML Telegramme von allen oder ausgewählten Verbindungen online mit dem Schema prüfen und validieren. Telegramme können pro Verbindung aufgezeichnet und mit einem XML Editor ausgewertet werden. Die Log-Datei enthält ein umhüllendes XML Element <...> zur Kennung:

<....><**X0 t**="2007-07-19T11:02:33.679"><**P n**="name"><**E v**="value" ...></**P**>...</**X0**><...>

|

<**X0**> = DxMonitor Read Telegram <X0> (Rückmeldung von DxNode an DxMonitor)
<**Y0**> = DxMonitor Write Telegram <X0> (Befehl vom DxMonitor an DxNode)

|

<**R0**> = DxNode Data Receive Telegram <X0> (Lesen von Partnerknoten/Anbindung)
<**S0**> = DxNode Data Send Telegram <X0> (Schreiben an Partnerknoten/Anbindung)

|

<**E1**> = DxNode Failure Message
<**E2**> = DxNode Warning Message
<**E4**> = DxNode Information Message

|

Umhüllendes XML Element zur Darstellung in der Log-Datei

Beispiel aus DxMonitor Log-Datei zum prüfen der Telegramme zwischen DxNode und Anbindung:

XML Telegramme DxNode⇔TestEm mit DxMonitor beobachtet

Bemerkung: zur besseren Übersicht wurden die Werte für Zeitstempel **t**=".." entfernt

DxMonitor Kontrollabfrage der Datenpunkte (Ausgabe q="bWD" bedeutet Quality="badWaitingForData")

```
<Y0 t=".."><SX><P a="EM01*" r="=" w="="/></SX></Y0>
<X0 t="..">
    <SXR>
        <P a="EM01.clrPar"><D t=".." u="0" q="bWD"/></P>
        <P a="EM01.incPar"><D t=".." u="0" q="bWD"/></P>
        <P a="EM01.sleepPar"><D t=".." u="0" q="bWD"/></P>
        <P a="EM01.val1"><D t=".." u="0" q="bWD"/></P>
        <P a="EM01.val2"><D t=".." u="0" q="bWD"/></P>
    </SXR>
</X0>
```

DxMonitor schaltet Verbose Mode auf maximal (alle Flags ON)

```
<Y0><Connect verbose="65535"/></Y0>
```

DxMonitor ist jetzt bereit für DxNode Überwachung:

TestEM startet und nimmt TCP/IP Verbindung mit DxNode auf

```
<E4 t=".." msg="new connect; Now active connects=4"/>
```

DxNode TCP/IP Verbindung wird angenommen und mit Socket-Handle bestätigt

```
<E4 t=".." msg="new_client [127.0.0.1] on socket: 1724"/>
```

TestEM fordert die Vermittlung mit Connect Name cn="Conn01" an

```
<R0 t=".."><X0><Connect cn="Conn01"><Switch/></Connect></X0></R0>
```

DxNode bestätigt die Vermittlung und schaltet die Verbindung auf cn="Conn01"

```
<S0 t=".."><X0><ConnectR cn="Conn01" /></X0></S0>
<E4 t=".." msg="connect successfully switched; Now active connects=3"/>
<E4 t=".." msg="*** Link Initialize ***"/>
<E4 t=".." msg="Connect Conn01 connected"/>
```

DxNode sendet Server Matrix <SX> als Generalabfrage

```
<S0 t=".."><SX><P a="*Par" w="=" r="="/><P a="*Val*" r="="/></SX></S0>
```

TestEM sendet alle Werte zur Initialisierung

```
<R0 t="..">
    <X0 t="..">
        <SXR>
            <P a="EM01.incPar"><D v="1"/></P>
            <P a="EM01.sleepPar"><D v="1000"/></P>
            <P a="EM01.val1"><D v="0"/></P>
            <P a="EM01.val2"><D v="0"/></P>
        </SXR>
    </X0>
</R0>
```

☞ <**SXR**> kann auch weggelassen werden, es bedeutet Anfang <**SXR**> und Ende </**SXR**> der Initialisierung.
Daten <**D**> können auch als Event <**E**> übermittelt werden.

TestEM sendet spontan Änderungen <P>

```
<R0 t=".."><X0 t=".."><P a="EM01.val1"><E v="1"/></P><P a="EM01.val2"><E v="1"/></P></X0></R0>
<R0 t=".."><X0 t=".."><P a="EM01.val1"><E v="2"/></P><P a="EM01.val2"><E v="2"/></P></X0></R0>
<R0 t=".."><X0 t=".."><P a="EM01.val1"><E v="3"/></P><P a="EM01.val2"><E v="3"/></P></X0></R0>
```

Die Verbindung cn="Conn01" wurde terminiert z.B. durch Abschalten von TestEM

```
<E4 t=".." msg="*** Link Shutdown ***"/>
<E4 t=".." msg="Connect Conn01 standby"/>
```

DxMonitor schaltet Verbose Mode aus (alle Flags OFF)

```
<Y0><Connect verbose="0"/></Y0>
```

DxMonitor Kontrollabfrage der Datenpunkte (Ausgabe q="bCF" bedeutet Quality="badCommFailure")

```
<Y0 t=".."><SX><P a="EM01*" r="=" w="="/></SX></Y0>
<X0 t="..">
    <SXR>
        <P a="EM01.clrPar"><D t=".." u="0" q="bCF"/></P>
        <P a="EM01.incPar"><D v="1" t=".." u="0" q="bCF"/></P>
        <P a="EM01.sleepPar"><D v="1000" t=".." u="0" q="bCF"/></P>
        <P a="EM01.val1"><D v="1" t=".." u="0" q="bCF"/></P>
        <P a="EM01.val2"><D v="1" t=".." u="0" q="bCF"/></P>
    </SXR>
</X0>
```

Transaktionen

Alle Transaktionen basieren auf **Stream Sockets** und **TCP**/**IP**, das so genannte "Reliable Transactions" mit PAR (Positive Acknowledge with Re-transmission) garantiert. Jede Verbindung (auch lokale) werden kontinuierlich mit einem **Alive Signal** überwacht. Dies erlaubt eine schnellere Fehlererkennung, als dies mit den Einstellungen von TCP/IP möglich ist. Eine Sicherheit bietet das **Speichern und Weiterleiten** (Store&Fwd) von Meldungen das pro Verbindung und Datenpunkt parametriert werden kann.

Für den Datenaustausch müssen Client (Datenanwender) und Server (Datenerbringer) definiert sein. Dies erfolgt mit der Parametrierung der Client <**CX**> und Server <**SX**> Matrix pro Verbindung <**Connect**>. Grundsätzlich kennt DxNode nur zwei Transaktionen, Lesen und Schreiben vom/zum Server, wobei diese im Telegramm markiert sind: <**E**.../> für **Lesen** vom Server und <**e**.../> für **Schreiben** zum Server. Zur Erläuterung sei der Datenverkehr zwischen Prozess (Server) und SCADA (Client) herangezogen:

Lesen (Read from Server) bedeutet hier Daten von **Prozess→SCADA** (Client) zu übermitteln. Alle zu lesenden Daten werden nach erfolgreichem Verbindungsaufbau im Empfänger Knoten zuerst abgeglichen d.h. als Generalabfrage <**SXR**> gelesen (Synchronisation). Nach der Synchronisation sind zwei Transaktionstypen möglich:

- **Einfaches Lesen** <**E v**=".." **t**=".."/> vom Server ist die Standard Transaktion zum übermitteln von Prozessdaten. Dabei können alle oder nur geänderte Attributwerte übertragen werden. Fehlende Attribute werden in DxNode automatisch ergänzt z.B. wird der Zeitstempel **t**=".." erzeugt oder die Quality wird **q**="g" (good) gesetzt.
- **Indizierte Nachricht** <**E v**=".." **t**=".." **i**="##"/> mit **Rückmeldung** <**e t**=".." **i**="##"/> wobei ## ein im Server definierter Index darstellt der mindestens von einem zuständigen Client zurückgemeldet werden muss. Diese Transaktion kann verwendet werden um Meldungen explizit zu bestätigen, sie benötigt jedoch eine entsprechende Applikation in Client und Server.

Schreiben (Write to Server) bedeutet hier Daten von **SCADA→Prozess** (Server) übermitteln. Zu schreibende Daten werden nach erfolgreichem Verbindungsaufbau nicht automatisch abgeglichen d.h. Befehle an einen Server werden nicht nachgeführt wenn die Verbindung unterbrochen wurde. Nach dem Verbindungsaufbau sind zwei Transaktionstypen möglich:

- **Einfaches Schreiben** <**e v**=".." **t**=".."/> an den Server kann verwendet werden um eine Meldung ohne Rückmeldung zu übermitteln.
- **Bi-direktionaler Befehl** <**e v**=".." **t**=".." **q**="gT"/> mit **Rückmeldung** <**E v**=".." **t**=".." **q**="g"/> ist die Standard Transaktion für Steuerbefehle. Dabei werden die Daten normalerweise vom Server gelesen (Read from Server) und nur für die Befehlsaktion beschrieben (Write to Server). Ein neuer Wert wird mit Quality **q**="gT" (good in Transition) im Server empfangen und muss zurückgemeldet werden. Die Quality muss dabei nicht übermittelt werden, sie wird - wie beim Lesen – von DxNode automatisch auf **q**="g" (good) gesetzt. Der zurückgemeldete Wert kann, muss aber nicht den Befehlswert darstellen. Zum Beispiel kann der Befehl "Motor start" mit Wert **v**="3" beantragt und die Rückmeldung "Motor läuft" mit **v**="2" angezeigt werden. Damit lassen sich Anlagenzustände in einem DP Wert (Enum) abbilden. Andererseits kann ein Sollwert z.B. mit **v**="17.5" beantragt und 1:1 zurückgemeldet werden. Befehlswert und Rückmeldung können in der Prozessebene unterschiedliche Adressen haben, wenngleich die Werte im Netzwerk nur einen einzigen Datenpunkt darstellen.

Zeitstempel und Quality

Zeitstempel **t**=".." sind nach XML gemäss **ISO-8601** YYYY-MM-DDThh:mm:ss.xx in [ms] formatiert. Sie sollen zeitfolgerichtig übermittelt werden, wenn nicht, dann werden sie ggf. in DxNode mit dem Attribut **tc**=".." für eine Zeitstempelkorrektur markiert. Fehlende Zeitstempel und Quality Attribute werden im ersten DxNode automatisch ergänzt d.h. der Zeitstempel **t**=".." wird dann von DxNode selbst erzeugt oder die Quality wird **q**="g" (good) gesetzt. Bei Verbindungsunterbruch zur Prozessebene kann die Quality automatisch z.B. **q**="b" (bad) gesetzt werden (parametrierbar mit <**LinkOff**>).

Dateninhalte und Datentypen

DxNode dient der Standardisierung und der zuverlässigen Übertragung von Ereignissen, also nicht primär der Definition von Dateninhalten. Diese werden pro Objekt an anderer Stelle vorgegeben und vorteilhaft für das gesamte Netzwerk normiert z.B. Bi-direktionaler Befehl mit Wert **v**="5" für Klappe ÖFFNEN, **v**="4" für Rückmeldung OFFEN. Die Inhalte sollten vom Anwender normiert werden, so ist gewährleistet, dass sich alle Teilnehmer verstehen. Es ist nicht sinnvoll, dass jede Anbindung eine eigene Methode zum Befehlen und Rückmelden verwendet. Deshalb sollten die Signale pro Anbindung entsprechend umgesetzt werden z.B. um statische Signale in Impulse zu wandeln usw. Wenn dies das Programm der Prozessebene nicht bereits vorsieht, dann muss es mit der Anbindung erfolgen.

Zeitstempel sollen zeitfolgerichtig übermittelt werden, wenn nicht, dann werden sie in DxNode mit einem zusätzlichen Attribut für eine Zeitstempelkorrektur **tc**=".." markiert. Fehlende Zeitstempel und Quality Attribute werden im ersten DxNode automatisch ergänzt d.h. der Zeitstempel **t**=".." wird dann von DxNode erzeugt oder die Quality wird **q**="g" (good) gesetzt. Bei Verbindungsunterbruch kann die Quality z.B. automatisch **q**="b" (bad) gesetzt werden (parametrierbar).

Signalkonditionierung

Grundsätzlich sind die Signale bereits in der externen Anlage aufzubereiten, so dass keine unnötigen Übertragungen ausgelöst werden nur weil z.B. ein Kontakt flattert. D.h. alle Signale werden als "Ereignisse" behandelt, die für das empfangende System eine klare Bedeutung haben z.B. ein Messwert hat sich eindeutig erhöht. Es ist daher vorgesehen, dass die Parametrierung zur Signalkonditionierung via DxNode an die Prozessebene weitergeleitet werden kann.

Die Parametrierung enthält alles was DxNode braucht, um autark zu laufen. Standardmässig sind zwei Adressräume für Signale vorgesehen und zwar für den Namen im Netzwerk mit **n**=".." sowie für eine Adresse **a**=".." als proprietäre Bezeichnung. Elementdaten <**E**> können z.B. für Datentyp+Format **f**=".." angefügt werden. Zusätzlich kann die <**DPList**> pro Datenpunkt <**P**> mit Element <**C**> für die Signalkonditionierung erweitert werden:

```
<DPList>
    <P n="Name1" a="Adr1"><E f="%typ1"/><C x=".." y=".." z=".." [..]/></P>
    <P n="Name2" a="Adr2"><E f="%typ2"/><C x=".." y=".." z=".." [..]/></P>
    <P n="NameX" a="AdrX"><E f="%typX"/><C x=".." y=".." z=".." [..]/></P>
</DPList>
```

Dabei werden unter <**C**> diejenigen Attribute **x,y,z** und Blätter definiert, die zur Signalaufbereitung notwendig sind aber von DxNode nicht direkt verwendet werden.

☞ Die Konfigurationsdaten <**C**> werden von DxNode nicht verarbeitet, sondern nur der Anbindung zur Verfügung gestellt.

Store & Forward Meldungen

Mit der Funktion Speichern und Weiterleiten (Store&Forward) kann DxNode ausgewählte Ereignisdaten auf die Harddisk auslagern und später übermitteln. Die Daten werden im XML Format in einen Ringpuffer geschrieben und bei Widerherstellung der Verbindung übermittelt. Die Konfiguration hierzu geschieht im ausschliesslich im Serverknoten mittels Attributen **store_fwd_buffer**="Store&Fwd Buffer" für die Puffergrösse und **attr**="S" in der Server Subscription <**SX**> für die gewünschten Datenpunkte einer Verbindung <**Connect**>. Der Grund dafür ist, dass der Server die Daten ja auch dann zwischenspeichern muss, wenn keine Verbindung besteht. Wenn der Serverknoten die Verbindung nicht herstellt (passiv), dann braucht es eine beidseitige Parametrierung: Die zu speichernden Datenpunkte werden mit <**Connect cn**=".."><**SX attr**="S"><**P**> im Server festgelegt und die Verbindung wird aus dem Client mit <**Connect cn**=".." **host**=".." **port**="."><**Switch**/> hergestellt. Dies ist z.B. für redundante Verbindungen erforderlich, wenn die Rechnerumschaltung im Client (aktiv) vorgenommen wird.

☞ Die Store&Fwd Definition **attr**="S" wirkt nur in der lokalen Server Matrix <**SX**>, eine Speicherung und Weiterleitung von Client-Daten (Befehle) ist nicht sinnvoll.

Redundante Anbindungen

DxNode unterstützt redundante Prozesse, indem zwei gleichwertige DxNode auf zwei separaten Rechnern aktiv/passiv oder parallel betrieben werden können. Voraussetzung dafür sind zwei Rechner mit eigenen IP-Adressen wobei auf jedem Rechner ein DxNode läuft um mit dem jeweils lokal installierten Prozessanbindungs-Programm zu kommunizieren. Es wird davon ausgegangen, dass die Daten redundanter Prozesse von deren proprietären Programmen und nicht via DxNode synchronisiert werden. Grundsätzlich unterstützt das Konzept zwei Methoden für redundante Anbindungen:

- **Beide DxNode laufen, nur einer wird aktiv mit der SCADA verbunden**, der zweite wird passiv verbunden. Allerdings sollen alle Ereignisse auch an den passiven DxNode übermittelt werden, damit er die Vorgeschichte aufzeichnet und sie bei einem allfälligen Unterbruch nach einer Umschaltung an die SCADA weiterleiten kann (Store&Fwd).
- **Beide DxNode laufen parallel**, d.h. alle Signale werden auf beide DxNode in beide Richtungen (Read und Write) 1:1 übertragen. In diesem Falle verwirft der zusammenfassende DxNode für die SCADA identische Telegramme. Diese sind dadurch gekennzeichnet, dass alle Attribute incl. Zeitstempel identisch sind d.h. die Methode ist sinnvoll, wenn die Zeitstempel zwischen den zwei Systemen abgeglichen sind. Bemerkung: Die Methode funktioniert grundsätzlich auch ohne Zeitstempelabgleich, d.h. wenn die Zeitstempel verschieden sind. Allerdings werden dann die Ereignisse doppelt, ggf. mit Zeitstempelkorrektur an die SCADA übertragen

XML Standard

Alle **Konfigurationsdateien** sollen die XML-Version und den Zeichencode als einheitliche Processing Instruction führen, z.B. ist <?**xml version**="1.0" **encoding**="ISO-8859-1"?> sinnvoll für Europa.

Zur Darstellung von **Daten** sind gemäss ➲ http://www.w3.org/XML nur die ASCII-Zeichen #x9 [TAB] (Tabulator), #xA [$^{L}_{F}$] (Line Feed), #xD [$^{C}_{R}$] (Carriage Return) und die darstellbaren Zeichen #x20..#x7F zulässig sowie alle weiteren Zeichen des jeweils netzweit gültigen Zeichensatzes.

Der XML Standard-Zeichensatz ist **Unicode UTF-8** ➲ http://www.unicode.org der sprachunabhängig alle Zeichen kennt und von DxNode unterstützt wird, aber eine entsprechende Umwandlung in der Anbindung erfordert, da die meisten Anwendungen in Europa den Zeichensatz **ISO-8859-1** verwenden.

DxNode ist **sprachunabhängig** und **unterstützt alle Zeichensätze** sofern sie XML-konform sind (genau genommen wird die Encoding Instruction ignoriert). Die XML-Telegramme brauchen daher keine Processing Instruction, wie oben dargestellt, was die zu übertragende Datenmenge reduziert. Der Zeichensatz ist also nur in den Konfigurationsdateien und in den Anwendungen zu berücksichtigen. Wenn alle Anwendungen mit dem gleichen Zeichensatz arbeiten, kann die Encoding Instruction weggelassen werden.

Eigennamen

Die Namen für Knoten **nn**="[NodeName]", Zugangsport **dn**="[DaemonName]" und Verbindungen **cn**="[ConnectName]" werden zur Adressbildung von Datenpunkten verwendet und sollen der **IEC-1131** Norm genügen. Deshalb sollen diese Namen nur alpha-numerische ASCII-Zeichen und Underscore (_) enthalten. Dies gilt im Wesentlichen auch für Host und Port Namen.

Wildcards

Wildcards oder so genannte Suchmasken dienen der Mehrfachauswahl bzw. Mehrfachzuweisung von Datenpunkten. **DxNode ist für Wildcards optimiert**, dafür sind zwei Zeichen reserviert:

? **Fragezeichen** (question mark) bedeutet "**EIN beliebiges Zeichen an dieser Stelle**"
***** **Sternchen** (asterisk) bedeutet "**eine beliebige Anzahl beliebiger Zeichen an dieser Stelle**"

Suchmasken sind "case sensitive" d.h. die Gross-/Kleinschreibung wird beachtet. DxNode unterstützt mehrere Fragezeichen und Sternchen in einer Suchmaske. Der Ausdruck **n**="*io*" referenziert z.B. alle Datenpunkte mit "io" irgendwo im Namen wie zum Beispiel "ABCio", "ioXYZ", "AioB", "Temp_io.val" etc. hingegen trifft er nicht "IoiOIO". Der Ausdruck **n**="io*" trifft z.B. nur "ioXYZ" aus obigen Beispielen.

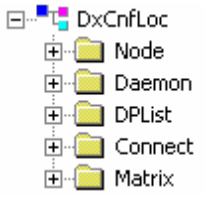
Konfiguration

DxNode benötigt ausser dem Standardprogramm **DxNode.exe** (Executable) eine in XML dargestellte Konfigurationsdatei z.B. **Node01.xml** (Parametrierung), die mindestens die zum Starten relevanten Informationen enthält. Auf diese, so genannte lokale Konfigurationsdatei ist mit dem Systemaufruf beim Start zu verweisen (Argument **-app** startet DxNode als Applikation, ➲ Anhang: DxNode Installation) ...

C:\Programme\DxNode\bin**DxNode.exe** X:\Konfiguration**Node01.xml -app**

Dabei sind Programm-, Konfigurations- und Arbeitsverzeichnis (siehe auch ➲ **path**="..") frei wählbar. Wenn kein Konfigurationsverzeichnis festgelegt wird, erwartet DxNode die Konfigurationsdatei im Arbeitsverzeichnis d.h. in dem Verzeichnis aus dem DxNode mit dem Systemaufruf gestartet wird.

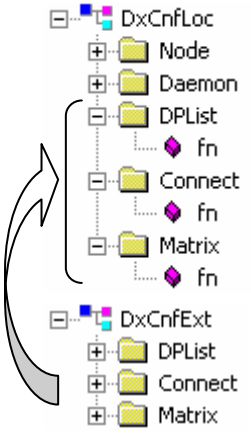
Die beim Starten zu übergebende Konfigurationsdatei beginnt mit dem Root Element <**DxCnfLoc**> und besteht aus fünf Hauptelementen, die alle Parametrierdaten für DxNode enthalten können:



Root Element	Konfiguration Lokal ...
<**Node**>	**Grundeinstellungen** von DxNode (erforderlich)
<**Daemon**>	**Zugangsport** für externe Teilnehmer (bedingt erforderlich)
<**DPList**>	**Datenpunktliste** mit Gruppeneinteilung (bedingt erforderlich)
<**Connect**>	**Anschluss/Verbindung** zu anderem Teilnehmer (optional)
<**Matrix**>	**Datenaustauschtabelle** für Verbindungen (optional)

- Das Element <**Node**> ist erforderlich, über den <**Daemon**> können andere Teilnehmer mit DxNode kommunizieren, er unterstützt alle Dienste, die für einen Datenaustausch notwendig sind. Die restlichen Elemente können lokal oder extern definiert werden.
- Die Minimalkonfiguration für einen DxNode mit lokalem Prozessabbild erfordert eine <**DPList**>, die das Prozessabbild darstellt (☞ Datenpunkte können jedoch mit **config_level**="1" auch automatisch erzeugt werden). DxNode kann so bereits über den <**Daemon**> von anderen Teilnehmern mit Daten beschickt und abgefragt werden.
- Ein <**Connect**> ist nur erforderlich, wenn DxNode selbst aktiv eine Verbindung zu einem anderen Teilnehmer herstellen - oder passiv, lokal definierte Daten zur Verfügung stellen soll. Pro Verbindung ist ein Element <**Connect**> in mindestens einem der beiden Teilnehmer zu definieren. Es können aber auch beide Teilnehmer ihren Teil der Verbindung festlegen und daher ein Element <**Connect**> haben. Dies ist z.B. dann erforderlich, wenn der eine Teilnehmer nur die Verbindung herstellen - der andere jedoch die auszutauschenden Daten definieren soll.
- Die <**Matrix**> bietet die Möglichkeit, alle auszutauschenden Daten pro Verbindung separat zu definieren. Der Inhalt der <**Matrix**> kann auch direkt im Element <**Connect**> definiert werden d.h. die <**Matrix**> stellt eine Untermenge der Verbindung ohne Verbindungseigenschaften dar. Der Vorteil einer <**Matrix**> liegt darin, dass die Datenaustauschtabellen neu definert werden können, ohne die Verbindungseigenschaften zu tangieren.

Genau genommen genügte bereits die Konfiguration von <**Node**> und <**Daemon**>, die restliche Parametrierung kann dann über eine Verbindung übermittelt werden. Damit ist der Knoten aber nicht autark, weil die externe Konfiguration z.B. nach einem Netzunterbruch nicht mehr verfügbar wäre.



In der Praxis empfiehlt sich eine **lokale Konfigurationsdatei** die einen autarken Betrieb von DxNode ermöglicht. Dabei kann für extern konfigurierbare Elemente <**DPList**>, <**Connect**> und <**Matrix**> eine externe Datei **fn**="[FileName]" referenziert werden, die - einmal heruntergeladen - auch nach einem Netzunterbruch zur Verfügung steht und damit DxNode unabhängig macht.

Die externe Datei **fn**="[FileName]" kann für alle Elemente die gleiche sein und wird beim Hochfahren von DxNode automatisch eingelesen. Für eine Online Parametrierung kann zusätzlich eine Verbindung mit den Funktionen <**GetFile**> und <**Include**> vorgesehen werden.

Die entsprechende **externe Konfigurationsdatei** hat exakt die gleiche Struktur wie der zu ersetzende Teil einer lokalen Konfiguration. Dabei können die Elemente <**DPList**>, <**Connect**> und <**Matrix**> den Knotennamen **nn**="[NodeName]" zur eindeutigen Zuordnung führen.

Für weitere Informationen siehe auch ➲ Beispiele Lokale und Externe Konfiguration.

Darstellungskonventionen

XML Elemente

Die folgenden Kapitel beschreiben alle Elemente mit den gültigen Attributen wie folgt in **Tabellenform**:

Root Element	Type	Range [Unit]	Use	Def
Simple Element	Elem	[text]	..	..
Complex Element	Elem	-	..	..
Attribut	%..	[data]	..	..

Die XML-Struktur entspricht der Darstellung von Microsoft™ XML Notepad. In allen Tabellen gilt für die Spalten **Type** (Datentyp), **Use** (Verwendung) und **Def** (Definition) folgende Legende:

Type	Elem=Simple Element vom Typ "string" bzw. Complex Element (grau) mit Attribut(en) vom Typ %s=string, %X=hexBinary, %b=boolean, %f oder %E =double/float/real, %i=int/integer, %L=enumerated/level, %y=date, %h=time, %yh=dateTime, siehe auch ➲ Attribut **f**="Format and Datatype"
Use	req=erforderlich (Attribut gelb, für externe Referenz blau), min1=mindestens ein Attribut erforderlich, xor=genau ein Attribut erforderlich, opt=optional, ⇳=steuerbar (rd/wr) oder ⇧=lesbar (rd) zur Laufzeit
Def	1✓=Element bzw. Attribut hier nur einmal definierbar, x✓=Element mehrere Male definierbar, 1ref=Attribut hier nur einmal referenzierbar d.h. es ist anderswo definiert

XML Attribute

Für die Detailbeschreibung der Attribute sei generell auf Kapitel ➲ Attribut Definition und Verwendung verwiesen. Dort würde z.B. die oben aufgeführte XML-Struktur in **Textform** wie folgt dargestellt:

```
<?xml version="1.0" encoding=".."?>
<DxCnfLoc>
   <Elem [text]/>
   <Elem Attr="[data]" />
</DxCnfLoc>
```

Die XML-Version und der Zeichensatz (Encoding) sind hier nur in der Textform sichtbar. Sie sollen in allen Konfigurationsdateien mit der gleichen Processing Instruction <?**xml** ...?> netzwerkweit einheitlich festgelegt werden.

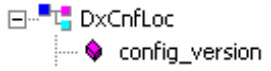
Die Textform erlaubt eine formelle Beschreibung der Einträge mit eigens dafür bestimmten Regeln. Dabei wird z.B. das jeweils im Kapitel beschriebene Attribut gelb hervorgehoben. Für alle in Textform dargestellten Elemente, Attribute und Werte gilt folgende Legende:

Darstellung	Beschreibung
[..]	Ausdruck oder Option mit unbestimmtem Inhalt
[x|y] bzw. a|b|c	Ausdruck oder Option "x" ODER "y" bzw. Auswahl "a" ODER "b" ODER "c"
[a-z0-9]	Ausdruck, Definition mit Auflistung zulässiger ASCII-Zeichen
[0..255]	Ausdruck, Definition mit Bereichsangabe
[NodeName]	Ausdruck referenziert d.h. an anderer Stelle oder durch Beschreibung definiert
<**X0**>	XML Start Tag oder referenziertes XML Element in Beschreibung
</**X0**> bzw. />	XML End Tag bzw. XML Element Abschluss
<**Switch**/>	XML Steuer-Element leer, so genannte Leer Tag
[<**E** .. />]	XML Element optional mit unbestimmtem Inhalt
[**a**=".."]	Attribut optional mit unbestimmtem Inhalt
a=".."	Attribut erforderlich mit unbestimmtem Inhalt
<**P n**=".." [..]/>	Attribut-Beschrieb **n**=".." gilt für <**P**> in jeder Hierarchieebene
<**X0**><**P a**|**n**="..">	Attribut-Beschrieb **a**|**n**=".." gilt nur für <**P**> in Telegramm <**X0**><**P**>
<**DPList**> <**Group gn**=".."> <**P a**=".." [..]/>	Attribut-Beschrieb **a**=".." gilt nur für <**P**> in <**DPList**><**Group**> Einzeilige Ersatzdarstellung: <**DPList**><**Group gn**=".."><**P a**=".." [..]/>
<**CX**> oder <**SX**> <**P n**=".." **r**=".." [..]/>	Attribut-Beschrieb **r**=".." gilt nur für <**P**> in <**CX**> ODER <**SX**> Einzeilige Ersatzdarstellung: <**CX**> oder <**SX**><**P n**=".." **r**=".." [..]/>
<**CX gn**="[X]"> <**P n**=".." [..]/> <**P gn**="[A]" **n**=".." [..]/>	Attribut-Überparametrierung: Attribute werden an die Elemente "vererbt" wenn sie dort nicht explizit aufgeführt sind. Zum Beispiel gilt hier **gn**="[X]" für alle Datenpunkte <**P**> ausser demjenigen der mit **gn**="[A]" überparametriert wurde.

Lokale Konfiguration <DxCnfLoc>

Eine lokale Konfigurationsdatei mit Root Element <**DxCnfLoc**> als Kennung ist **immer erforderlich**. Alle Konfigurationsdateien sollen die XML-Version und den Zeichensatz als einheitliche Processing Instruction führen, z.B. <?**xml version**="1.0" **encoding**="ISO-8859-1"?>. Zur Identifikation der Daten kann eine Versionsbezeichnung **config_version**="" angefügt werden:

<?**xml version**="" **encoding**=""?>

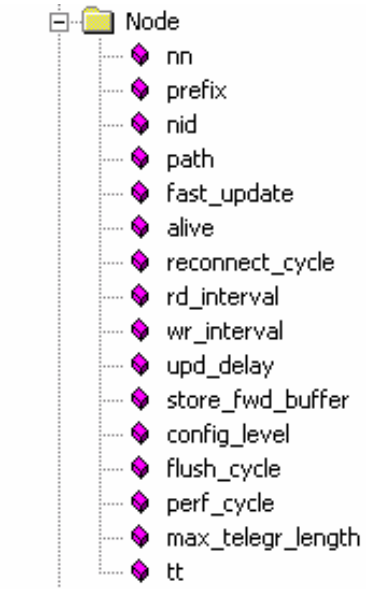


XML Processing Instruction	Type	Range [Unit]	Use	Def
Konfiguration Lokal	Elem	-	req	1 ✔
Configuration Version	%s	[text]	opt ⇧	1 ✔

Grundeinstellungen <Node>

Mit dem Element <**Node**> können für DxNode allgemein geltende Parameter wie Knotenname **nn**="" oder global "vererbbare" Attribute wie **alive**="" für die Verbindungsüberwachung etc. definiert werden. Der Knotenname wird auch in externen Konfigurationen referenziert oder dient als Bezeichner für interne Datenpunkte. Gültige Einträge sind:

<**DxCnfLoc**> Fortsetzung



Konfiguration Lokal	Type	Range [Unit]	Use	Def
Grundeinstellungen	Elem	-	req ⇳	1 ✔
Node Name	%s	[A-Za-z0-9_]	req	1 ✔
Internal DP Name Prefix	%s	[A-Za-z0-9_]	opt	1 ✔
Node Identification	%s	[text]	opt	1 ✔
Log File and Store&Fwd Path	%s	[text]	opt	1 ✔
Fast Update Control Flag	%b	false/true	opt ⇳	1 ✔
Alive Timeout **	%i	1..9999 [sec]	opt ⇧	1 ✔
Reconnect Time **	%i	1..65535 [sec]	opt ⇧	1 ✔
Read Interval **	%i	0..65535 [ms]	opt ⇧	1 ✔
Write Interval **	%i	0..65535 [ms]	opt ⇧	1 ✔
Update Delay **	%i	0..65535 [ms]	opt ⇧	1 ✔
Store&Fwd Buffer **	%i	0..100000 [KB]	opt	1 ✔
Auto Configuration **	%i	0=None, 1=DPList	opt	1 ✔
Store&Fwd Flush Cycle	%i	500..1000000 [ms]	opt	1 ✔
Performance Check Interval	%i	0=Off, 500..65535 [ms]	opt ⇳	1 ✔
Max Length of Telegram	%i	1..2097152 [KB]	opt	1 ✔
Time Tolerance	%i	0..65535 [ms]	opt	1 ✔

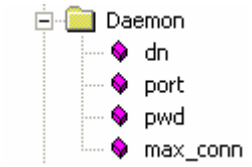
** diese Attribute können für jede Verbindung <**Connect**> individuell überparametriert werden.

DxNode erzeugt zur Laufzeit ➲ interne Datenpunkte mit Adressen **a**="[Prefix][NodeName]....". Dabei kann DxNode mittels [Prefix][NodeName].cmdio.state gesteuert werden (shutdown oder restart). Die mit ⇳ markierten Attribute sind online einstellbar, diejenigen mit ⇧ sind pro Verbindung online lesbar und können dort ggf. überparametriert werden.

Zugangsport <Daemon>

Element <**Daemon**> definiert ein Zugangsport für externe Anschlüsse bzw. Dienstanforderer die sich entweder mit Daemon Name **dn**="" oder **port**="" anmelden. Eine Portadresse bzw. die Service-Nr, ist für den Daemon Prozess erforderlich. Der Daemon Name wird auch referenziert oder dient als Bezeichner für interne Datenpunkte. Gültige Einträge sind:

<**DxCnfLoc**> Fortsetzung



Konfiguration Lokal	Type	Range [Unit]	Use	Def
Zugangsport	Elem	-	req ⇳	x ✔
Daemon Name	%s	[A-Za-z0-9_]	req	1 ✔
Port Addr or Service-Nr	%s	[text]	req	ref
Password for Connects	%s	[text]	opt	1 ✔
Max Number of Connects	%i	1..255 [number]	opt ⇧	1 ✔

DxNode erzeugt zur Laufzeit ➲ interne Datenpunkte mit Adressen **a**="[Prefix][DaemonName]...". Dabei kann der Daemon mittels [Prefix][DaemonName].cmdio.state gesteuert werden (shutdown oder stop/restart). Die mit ⇧ markierten Attribute sind online lesbar.

Der Daemon unterstützt sowohl nicht-parametrierte Verbindungen (z.B. für DxMonitor oder nur im Partnersystem definierte) wie auch parametrierte, auf welche nach der Verbindungsherstellung umgeschaltet werden kann. Der Daemon stellt normalerweise den einzigen Zugangspunkt an DxNode dar (es sind jedoch mehrere Daemone parametrierbar), er unterstützt ein "Listener Port" (Horcher) über das sich alle Teilnehmer anmelden müssen.

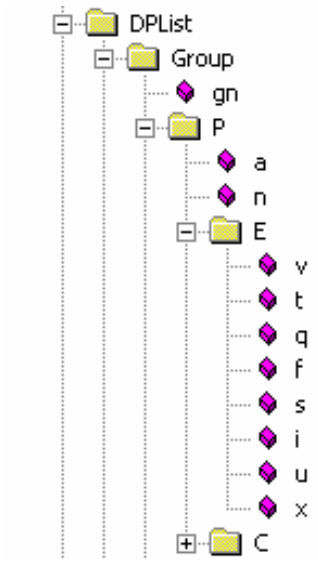
Der Zugangsport kann mit einem Passwort **pwd**="[Password] geschützt werden. Alle Teilnehmer müssen dieses Passwort für einen erfolgreichen Zugriff ebenfalls konfiguriert haben. Mit dem Zugriffsschutz des Ports kann DxNode auf einfache Weise vor unerwünschtem Zugang geschützt werden.

Innerhalb desselben Rechners ist für jeden <**Daemon**> eine eigener **port**=".." zu vergeben. Über diesen Port empfängt DxNode Anfragen von anderen Teilnehmern und führt entsprechende "Services" (Dienste) aus z.B. Subskriptionen <**CX**> oder <**SX**> (Dauerauftrag für Datenübermittlung).

Datenpunktliste <DPList>

Im Element <**DPList**> werden die Datenpunkte von DxNode definiert. Die Datenpunkte sollen mit den Elementen <**Group**> gruppiert werden. Mit den Elementen <**P**> werden die Datenpunkte definiert wobei jeder Datenpunkt zwei Adressräume belegen kann: Attribut **n**=".." definiert den netzweit gültigen Namen mit einem beliebigen Kennzeichnungssystem, Attribut **a**=".." enthält die proprietäre Adresse des angebundenen Systems und ggf. Elementdaten <**E**>. Gültige Einträge sind:

<**DxCnfLoc**> Fortsetzung



Konfiguration Lokal	Type	Range [Unit]	Use	Def
Datenpunktliste	Elem	-	opt	1
Gruppe	Elem	-	req	x
Group Name	%s	[A-Za-z0-9_]	req	1
Datenpunkt	Elem	-	req	x
Address (application)	%s	[A-Za-z0-9_]	min1	1
Name (network)	%s	[text]	min1	1
Elementdaten	Elem	-	opt	1
Value	%s	[text]	opt	1
Timestamp	%yh	[YMDhmsx]	opt	1
Quality	%s	[selection]	opt	1
Format and Datatype	%s	[selection]	opt	1
Status	%i	0..255	opt	1
Message Index	%i	±2147483647	opt	1
Engineering Unit	%s	[selection]	opt	1
Text Information	%s	[text]	opt	1
Konditionierungsdaten	Elem	-	opt	1

Element <**E**> enthält allfällige Standardeinstellungen die beim Start von DxNode übernommen werden. Element <**C**> enthält ➲ proprietäre Konditionierungsdaten, die von DxNode ignoriert werden.

Alternative zur Referenzierung einer extern parametrierten <**DPList**>:

<**DxCnfLoc**> Alternative

Konfiguration Lokal	Type	Range [Unit]	Use	Def
Datenpunktliste	Elem	-	opt	x
File Name	%s	[FileName]	req	1ref

In diesem Fall wird eine <**DPList**> aus der externen Konfigurationsdatei **fn**=".." eingefügt. Es können eine lokale und/oder mehrere externe <**DPList**> definiert werden. Eine extern definierte <**DPList**> kann zur Laufzeit mit dem <**Include**> Befehl eingefügt werden. Dabei werden nur neue Datenpunkte angehängt, bestehende können zur Laufzeit nicht verändert oder gelöscht werden.

Datenpunktgruppen <Group> und Datenpunkte <P>

Der Gruppenname **gn**="." wird allen, in der Baumstruktur untergeordneten Datenpunkten <**P**> zugeordnet wird. Ein Datenpunkt <**P**> darf nur einer Gruppe zugewiesen werden. Der Gruppenname wird als Auswahlkriterium mit Wildcards (*?) bei der Subskription von Datenpunkten referenziert, siehe auch <**CX**> und <**SX**> von <**Connect**> bzw. <**Matrix**>. Gruppen sollten daher Datenpunkte mit gleichen Eigenschaften bzgl. des Datenaustauschs enthalten, weil damit die Parametrierung der Auswahl vereinfacht wird. Kriterien sind z.B. Örtlichkeiten (Verbindungswege), Alarme (quittierbar), Zustände (nur lesbar), Befehle (bi-direktional), etc. Man kann so z.B. alle bi-direktionalen Datenpunkte für eine bestimmte Verbindung mit einem einzigen Eintrag in der Matrix definieren.

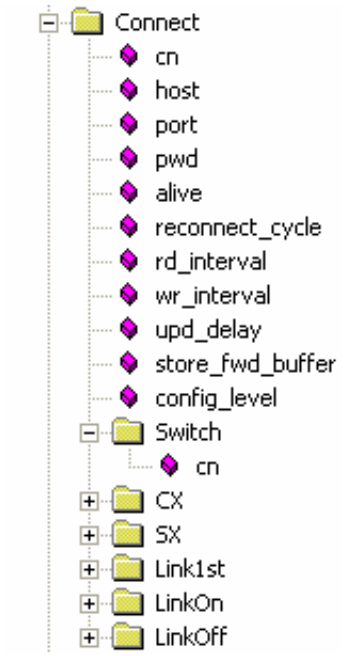
Eine andere Gruppierung ist ggf. sinnvoll, wenn das Kennzeichnungssystem die gewünschten Auswahlkriterien direkt liefert. Datenpunktenamen wie z.B. **n**="OrtA_AnlageB_ObjektC_TypD" können dank ihrer Struktur direkt in der Matrix mit Wildcards z.B. mit **n**="OrtA*TypD" gruppiert werden. Der Gruppenname **gn**="." ist also dann sinnvoll, wenn das Kennzeichnungssystem keine einfache Gruppierung der Datenpunkte ermöglicht.

Gruppen und/oder Wildcards wurden beim Design der Matrix-Funktion berücksichtigt - sie werden effizienter verarbeitet, als viele, explizit aufgeführte Datenpunkte. Eine mit Gruppen und/oder Wildcards bestückte Matrix ist zudem übersichtlich, da sie nur die wenigen Datenpunkte enthält, die explizit parametriert werden müssen. Das Ziel sollte daher sein, den wesentlichen Datenaustausch mit Gruppen und/oder Wildcards zu parametrieren und nur die Ausnahmen aufzulisten.

Anschluss/Verbindung <Connect>

Mit dem Element <**Connect**> wird ein Anschluss bzw. eine Verbindung zu einem anderen Partner definiert. Der Partner kann ein angekoppeltes System oder jeder andere DxNode sein. Die Minimal-Konfiguration erfordert den Verbindungsnamen **cn**="." zur Identifikation. Der Name wird vom Partner referenziert und dient als Bezeichner für ➲ interne Datenpunkte. Alle anderen Attribute und Elemente sind für die Verbindungsdefinition nicht zwingend. Das Attribut **alive**="." definiert eine individuelle Verbindungsüberwachungszeit. Die Attribute **host**="." und **port**="." definieren einen aktiven Anschluss zum Rechner mit Name **host**="." und Daemon **port**="." . Wenn diese Attribute fehlen, handelt es sich um einen passiven Anschluss, der von einem Partner über den lokalen Daemon angesprochen werden kann. Gültige Einträge sind:

<**DxCnfLoc**> Fortsetzung



Konfiguration Lokal	Type	Range [Unit]	Use	Def
Anschluss/Verbindung	Elem	-	opt ⇳	x ✔
Connect Name (local)	%s	[A-Za-z0-9_]	req	1 ✔
Host Name or IP-Address	%s	[text]	opt ⇳	1ref
Port Addr or Service-Nr	%s	[text]	opt ⇳	1ref
Password to Access Daemon	%s	[text]	opt	1 ✔
Alive Timeout	%i	1..9999 [sec]	opt ⇳	1 ✔
Reconnect Time	%i	1..65535 [sec]	opt ⇳	1 ✔
Read Interval	%i	0..65535 [ms]	opt ⇧	1 ✔
Write Interval	%i	0..65535 [ms]	opt ⇧	1 ✔
Update Delay	%i	0..65535 [ms]	opt ⇧	1 ✔
Store&Fwd Buffer	%i	0..100000 [KB]	opt	1 ✔
Auto Configuration	%i	0=None, 1=DPList	opt	1 ✔
Umschaltung/Vermittlung	Elem	-	opt	1 ✔
Connect Name (remote)	%s	[A-Za-z0-9_]	opt	1ref
Client Matrix/Subscription	Elem	-	opt ⇳	1 ✔
Server Matrix/Subscription	Elem	-	opt ⇳	1 ✔
Verbindung 1st Start-Up	Elem	-	opt	1 ✔
Verbindung ON	Elem	-	opt	1 ✔
Verbindung OFF	Elem	-	opt	1 ✔

Das Element <**SX**> unterstützt die gleichen Elemente und Attribute wie das Element <**CX**>.

DxNode erzeugt zur Laufzeit ➲ interne Datenpunkte mit Adressen **a**="[Prefix][ConnectName]...". Dabei kann die Verbindung mittels [Prefix][ConnectName].cmdio.state gesteuert werden (shutdown oder stop/restart). Die mit ⇳ markierten Attribute sowie die Elemente <**CX**> und <**SX**> sind online einstellbar, die Attribute mit ⇧ sind online lesbar.

Alternative zur Referenzierung einer extern parametrierten Verbindung <**Connect**>:

<**DxCnfLoc**> Alternative

Konfiguration Lokal	Type	Range [Unit]	Use	Def
Anschluss/Verbindung	Elem	-	opt	1 ✔
File Name	%s	[FileName]	req	1ref

In diesem Fall wird die Verbindung <**Connect**> aus der externen Konfigurationsdatei **fn**=".." eingefügt. Es können mehrere lokale und/oder externe <**Connect**> definiert werden. Eine Verbindung wird jedoch nur beim Hochfahren d.h. nicht zur Laufzeit eingefügt. Eine Verbindung kann also nur hier mit dem Dateinamen **fn**=".." referenziert, jedoch nicht mit dem <**Include**> Befehl eingefügt werden.

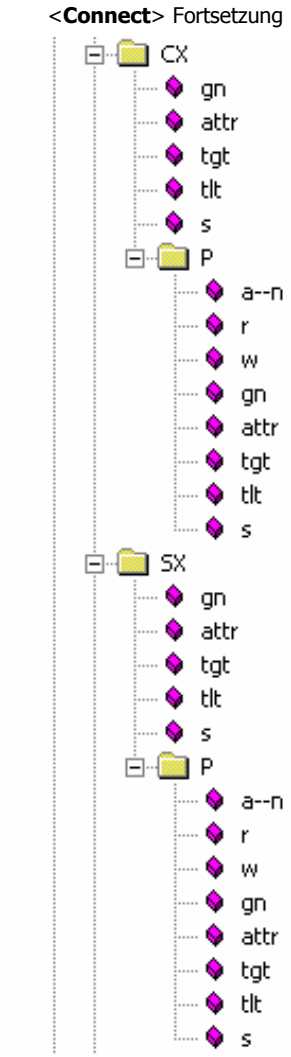
Aktive und Passive Anschlüsse

- Ein **aktiver Anschluss** ist dadurch gekennzeichnet, dass die Attribute **host**=".." und **port**=".." im Element <**Connect**> definiert sind. In diesem Falle versucht DxNode eine Verbindung zum Rechner mit Name/IP-Adresse **host**=".." und Zugangsport **port**=".." aktiv herzustellen. Eine aktive Verbindung benötigt im passiven Partnerknoten keine Parametrierung d.h. der Datenverkehr wird dort vollständig vom <**Daemon**> abgewickelt. Man spricht auch von einer nicht-parametrierten Verbindung im passiven Teil was den Vorteil unterstreicht, dass dort KEINE Konfiguration benötigt - und damit z.B. die Netzwerkparametrierung erleichtert wird. Über eine aktive Verbindung kann z.B. DxMonitor die Verbindungen von DxNode beobachten, ohne dafür vorher einen Anschluss reserviert zu haben.
- Ein **passiver Anschluss** kann (muss aber nicht) parametriert werden. Wenn die Parametrierung fehlt, wird der Datenverkehr vom <**Daemon**> im passiven Knoten abgehandelt. Andererseits werden für solche nicht-parametrierten Verbindungen auch keine ➲ internen Datenpunkte zur Verfügung gestellt, um den passiven Verbindungsteil zu steuern oder zu überwachen. Wenn dies gefordert ist, dann muss auch der passive Teil der Verbindung parametriert werden. Eine parametrierte Verbindung ist dadurch gekennzeichnet, dass mindestens das Element <**Connect**> mit dem Verbindungsnamen **cn**=".." definiert wird. Jede parametrierte Verbindung, kann gesteuert und überwacht werden, weil für solche Verbindungen immer ➲ interne Datenpunkte bestehen. Eine Anwendung davon ist z.B., dass eine Anlage mit einem passiven Knoten jederzeit den Verbindungszustand prüfen kann. Um eine Vermittlung zu einem parametrierten passiven Anschluss herzustellen, muss der aktive Teil die Verbindung mit dem Schaltelement <**Switch**> und dem gewünschten Namen **cn**=".." anfordern. Wenn der Name beidseitig gleich ist, braucht er nicht speziell parametriert bzw. übermittelt zu werden.
- Eine **beidseitige Parametrierung** ist erforderlich für Redundante Anbindungen, wenn Daten vom Server zwischengespeichert und bei Widerherstellung der Verbindung weitergeleitet werden sollen (Store&Fwd). In diesem Falle wird die Rechnerumschaltung im aktiven Client vorgenommen, z.B. wenn die Verbindung unterbrochen wird, während die Store&Fwd Parametrierung im passiven Server definiert werden muss, weil ja dieser auch dann die Daten zwischengespeichern soll wenn noch keine Verbindung zum Client besteht.

Subskriptionen <CX><SX>

Die Elemente <**CX**> und <**SX**> definieren eine **C**lient Matri**X** bzw. **S**erver Matri**X**, auch Subskription genannt. Normalerweise können <**CX**> und/oder <**SX**> im aktiven Partner konfiguriert werden. Dies erübrigt eine Konfiguration im passiven Partner Knoten. Allerdings sind für nicht-parametrierte Verbindungen im passiven Knoten dann auch keine ➲ internen Datenpunkte verfügbar.

<**Connect**> Fortsetzung



Anschluss/Verbindung	Type	Range [Unit]	Use	Def
Client Matrix/Subscription	Elem	-	opt ⇳	1 ✓
Group Name	%s	[GroupName|wildcard]	opt	1ref
Attributes	%s	[a,n,v,t,q,f,s,i,u,x,c]	opt	1 ✓
Time Greater Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Time Less Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Status Bit Filter	%s	[bbbbbbbb] (b=?,0,1)	opt	1 ✓
Datenpunkt-Auswahl	Elem	-	req	x ✓
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
Read from Server Addr|Name	%s	[Addr|Name|wildcard|=]	req	1ref
Write to Server Addr|Name	%s	[Addr|Name|wildcard|=]	opt	1ref
Group Name	%s	[GroupName|wildcard]	opt	1ref
Attributes	%s	[a,n,v,t,q,f,s,i,u,x,c]	opt	1 ✓
Time Greater Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Time Less Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Status Bit Filter	%s	[bbbbbbbb] (b=?,0,1)	opt	1 ✓
Server Matrix/Subscription	Elem	-	opt ⇳	1 ✓
Group Name	%s	[GroupName|wildcard]	opt	1ref
Attributes	%s	[S,a,n,v,t,q,f,s,i,u,x,c]	opt	1 ✓
Time Greater Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Time Less Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Status Bit Filter	%s	[bbbbbbbb] (b=?,0,1)	opt	1 ✓
Datenpunkt-Auswahl	Elem	-	req	x ✓
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
Read from Server Addr|Name	%s	[Addr|Name|wildcard|=]	req	1ref
Write to Server Addr|Name	%s	[Addr|Name|wildcard|=]	opt	1ref
Group Name	%s	[GroupName|wildcard]	opt	1ref
Attributes	%s	[S,a,n,v,t,q,f,s,i,u,x,c]	opt	1 ✓
Time Greater Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Time Less Than	%s	[YMDhmsx|wildcard]	opt	1 ✓
Status Bit Filter	%s	[bbbbbbbb] (b=?,0,1)	opt	1 ✓

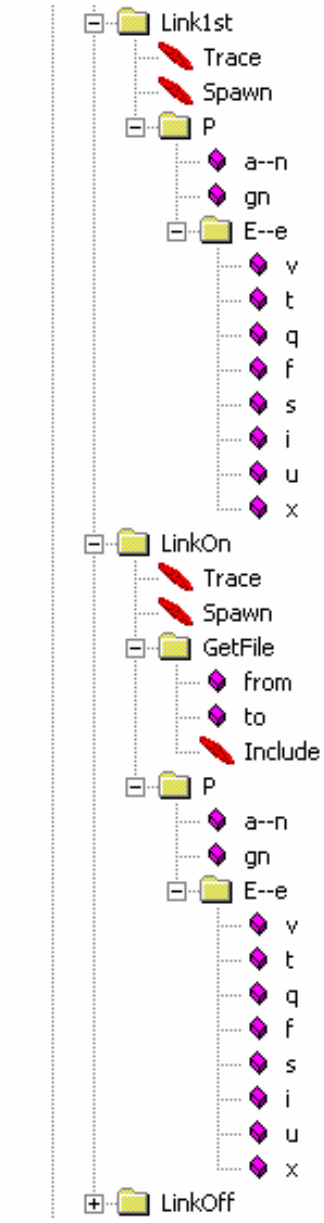
DxNode erzeugt zur Laufzeit ➲ interne Datenpunkte mit Adressen **a**="[Prefix][ConnectName]....". Dabei können Client - und Server-Matrix mittels [Prefix][ConnectName].matrix.cx bzw. matrix.sx online eingestellt werden.

☞ Die Store&Fwd Definition **attr**="S" wirkt nur in der lokalen Server Matrix <**SX**>, eine Speicherung und Weiterleitung von Client-Daten (Befehle) ist nicht sinnvoll.

Steuerelemente <Link1st> <LinkOn> <LinkOff>

Die Steuerelemente erlauben die Einstellung von Datenpunkten sowie die Ausführung von Befehlen beim erstmaligen Hochfahren mit <**Link1st**>, bei Verbindungs(wieder)herstellung mit <**LinkOn**> oder beim Abschalten bzw. bei Verlust der Verbindung mit <**LinkOff**>. Gültige Einträge sind:

<**Connect**> Fortsetzung



Anschluss/Verbindung	Type	Range [Unit]	Use	Def
Verbindung 1st Start-Up	Elem	-	opt	1
Log File Trace Message	Elem	[text]	opt	x
Program Start Instruktion	Elem	[FileName+Argument]	opt	x
Datenpunkt-Auswahl	Elem	-	opt	x
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
DP Group Name	%s	[GroupName|wildcard]	opt	1ref
Elementdaten	Elem	E=from e=to Server	opt	1
Value	%s	[text]	opt	1
Timestamp	%yh	[YMDhmsx]	opt	1
Quality	%s	[selection]	opt	1
Format and Datatype	%s	[selection]	opt	1
Status	%i	0..255	opt	1
Message Index	%i	±2147483647	opt	1
Engineering Unit	%s	[selection]	opt	1
Text Information	%s	[text]	opt	1
Verbindung ON	Elem	-	opt	1
Log File Trace Message	Elem	[text]	opt	x
Program Start Instruktion	Elem	[Program+Arguments]	opt	x
GetFile Instruktion	Elem	-	opt	x
Source File	%s	[SourceFileName]	req	1ref
Target File	%s	[TargetFileName]	req	1
Include File Instruktion	Elem	-	opt	1
Datenpunkt-Auswahl	Elem	-	opt	x
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
DP Group Name	%s	[GroupName|wildcard]	opt	1ref
Elementdaten	Elem	E=from e=to Server	opt	1
Value	%s	[text]	opt	1
Timestamp	%yh	[YMDhmsx]	opt	1
Quality	%s	[selection]	opt	1
Format and Datatype	%s	[selection]	opt	1
Status	%i	0..255	opt	1
Message Index	%i	±2147483647	opt	1
Engineering Unit	%s	[selection]	opt	1
Text Information	%s	[text]	opt	1
Verbindung OFF	Elem	-	opt	1

Das Element <**LinkOff**> unterstützt die gleichen Elemente und Attribute wie <**Link1st**>.

Mit <**Trace**> können Log-Informationen generiert -, mit <**Spawn**> Programme gestartet - und mit <**P**> Datenpunkte bzw. deren Attribute gesetzt werden. In <**LinkOn**> können zudem Konfigurationsdateien mit <**GetFile**> übertragen und mit <**Include**> eingefügt werden.

☞ Elementdaten werden mit <**E**> oder <**e**> unterschiedlich gesetzt.
DxNode interpretiert <**E**> als "Read from Server" und sendet Daten nur an alle angeschlossenen Clients. Umgekehrt wird <**e**> als "Write to Server" interpretiert, d.h. die Daten werden nur an angeschlossene Server übermittelt, dabei wird die Quality ggf. automatisch auf **q**="gT" (in Transition) gesetzt.

Befehlselemente <Trace> <Spawn> <GetFile> und <Include>

- Element <**Trace**> erzeugt eine Warnungsmeldung <**E2 t**=".." **msg**="[text]" /> (Error Level 2) in der Log-Datei [NodeName].log bzw. [NodeName].bak im Arbeitsverzeichnis (siehe auch ➲ **path**="..") von DxNode. Dabei ist [text] die mit dem Befehlselement eingegebene Klartextmeldung <**Trace**>[text]</**Trace**>. Weitere Error-Meldungen vom Typ <**E1**>, <**E2**> oder <**E4**> werden ggf. von DxNode selbst erzeugt.
- Element <**Spawn**> ermöglicht das Starten eines Programms. Dabei stellt [Program+Arguments] den gewünschten Systemaufruf incl. Programmargument(en) dar. Zum Beispiel könnte man mit <**Spawn**>x:\Programm\DxNode.exe x:\Konfiguration\Node02.xml</**Spawn**> einen zweiten DxNode mit der Konfigurationsdatei Node02.xml starten.
- Element <**GetFile**> ist nur in <**LinkOn**> erlaubt und ermöglicht das Herunterladen (Download) einer Datei vom Partnerknoten. Dabei stellt [SourceFileName] den Quellenpfad+Dateinamen im Partnerrechner – und [TargetFileName] den Zielpfad+Dateinamen im lokalen Rechner dar.

  ☞ Eine externe Datei wird nur in den lokalen Rechner kopiert wenn die Zieldatei [TargetFileName] noch nicht existiert oder wenn diese 3 [sec] älter oder jünger ist als die zu herunterladende Datei. Damit wird verhindert, dass eine Datei unnötig oft heruntergeladen wird.
  Der Zeitvergleich basiert auf der Dateispeicherungszeit, die wegen unterschiedlicher Auflösung bei verschiedenen Systemen (Windows/Unix) eine gewisse Toleranz erfordert.

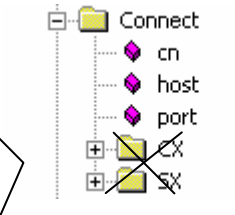
- Element <**Include**> fügt die in <**GetFile**> als [TargetFileName] bezeichnete externe Konfiguration in die lokale Konfiguration ein.

  ☞ Grundsätzlich werden zur Laufzeit nur Änderungen gegenüber der bestehenden (bereits laufenden) Konfiguration für die <**DPList**> und die <**Matrix**> aufgenommen. Dabei kann die <**DPList**> nur um Datenpunkte <**P**> erweitert - aber nicht aber reduziert werden. Andererseits kann eine Subskription <**CX**> und/oder <**SX**> nur ersetzt - aber nicht modifiziert werden.
  Mit dem Ersatz einer <**CX**> und/oder <**SX**> wird die entsprechende Verbindung <**Connect**> jeweils automatisch neu initialisiert, d.h. die Daten im Client werden neu auf den Server abgeglichen (Synchronisation).

Kommunikationsmatrix <Matrix>

Die <**Matrix**> bietet die Möglichkeit, alle auszutauschenden Daten pro Verbindung separat zu parametrieren. Sie stellt eine Untermenge einer Verbindung ohne Verbindungseigenschaften dar und enthält nur die Elemente <**CX**> und <**SX**>. D.h. diese Elemente entfallen dort und werden über den Verbindungsnamen **cn**=".." in der <**Matrix**> referenziert:

<**DxCnfLoc**> Fortsetzung



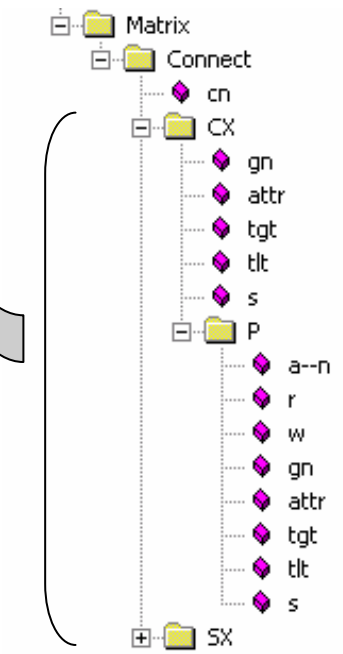
Konfiguration Lokal	Type	Range [Unit]	Use	Def
Anschluss/Verbindung	Elem	-	opt	x ✔
Connect Name	%s	[A-Za-z0-9_]	req	1 ✔
Host Name or IP-Address	%s	[text]	opt	1ref
Port Addr or Service-Nr	%s	[text]	opt	1ref

← Client/Server Matrix entfallen

In diesem Fall werden (ggf. definierte) <**CX**> und <**SX**> oben vollständig ersetzt und es gelten nur die in der <**Matrix**> unten definierten Client und/oder Server Subskriptionen.

Die Elemente <**CX**> und <**SX**> der <**Matrix**> definieren die zu kommunizierenden Datenpunkte. Die Kommunikationsmatrix ist also nichts anderes als eine Summe von Subskriptionen und erlaubt eine übersichtliche Darstellung für die referenzierten <**Connect**>. Gültige Einträge sind:

<**DxCnfLoc**> Fortsetzung



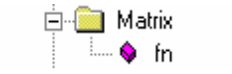
Konfiguration Lokal	Type	Range [Unit]	Use	Def
Kommunikationsmatrix	Elem	-	opt	1 ✔
Verbindungsreferenz	Elem	-	opt	x ✔
Connect Name	%s	[A-Za-z0-9_]	req	1ref
Client Matrix	Elem	-	opt	1 ✔
DP Group Name	%s	[GroupName|wildcard]	opt	1ref
Attributes	%s	[a,n,v,t,q,f,s,i,u,x,c]	opt	1 ✔
Time Greater Than	%s	[YMDhmsx|wildcard]	opt	1 ✔
Time Less Than	%s	[YMDhmsx|wildcard]	opt	1 ✔
Status Bit Filter	%s	[bbbbbbbb] (b=?,0,1)	opt	1 ✔
Datenpunkt-Auswahl	Elem	-	req	x ✔
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
Read from Server Addr|Name	%s	[Addr|Name|wildcard|=]	req	1ref
Write to Server Addr|Name	%s	[Addr|Name|wildcard|=]	opt	1ref
DP Group Name	%s	[GroupName|wildcard]	opt	1ref
Attributes	%s	[a,n,v,t,q,f,s,i,u,x,c]	opt	1 ✔
Time Greater Than	%s	[YMDhmsx|wildcard]	opt	1 ✔
Time Less Than	%s	[YMDhmsx|wildcard]	opt	1 ✔
Status Bit Filter	%s	[bbbbbbbb] (b=?,0,1)	opt	1 ✔
Server Matrix	Elem	-	opt	1 ✔

Das Element <**SX**> unterstützt die gleichen Elemente und Attribute wie das Element <**CX**>.

☞ Die Store&Fwd Definition mit **attr**="S" wirkt nur in der lokalen Server Matrix <**SX**>, eine Speicherung und Weiterleitung von Client-Daten (Befehle) ist nicht sinnvoll.

Alternative zur Referenzierung einer extern parametrierten <**Matrix**>:

<**DxCnfLoc**> Alternative



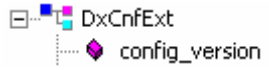
Konfiguration Lokal	Type	Range [Unit]	Use	Def
Kommunikationsmatrix	Elem	-	opt	1 ✔
File Name	%s	[FileName]	req	1ref

In diesem Fall wird eine <**Matrix**> aus der externen Konfigurationsdatei **fn**=".." eingefügt. Es können eine lokale und/oder mehrere externe <**Matrix**> definiert werden. Eine extern definierte <**Matrix**> kann zur Laufzeit mit dem <**Include**> Befehl eingefügt werden. Sie ersetzt dann die entsprechende aktuell laufende <**Matrix**> vollständig.

Externe Konfiguration <DxCnfExt>

Eine oder mehrere externe Konfigurationsdateien können mit Root Element <**DxCnfExt**> als Kennung definiert und mit dem Dateinamen **fn**=".." oder <**Include**> in der lokalen Konfiguration referenziert und eingefügt werden. Konfigurationsdateien sollen die XML-Version und den Zeichensatz als einheitliche Processing Instruction führen, z.B. <?**xml version**="1.0" **encoding**="ISO-8859-1"?>. Zur Identifikation der Daten kann pro Datei eine Versionsbezeichnung **config_version**=".." angefügt werden:

<?**xml version**=".." **encoding**=".."?>



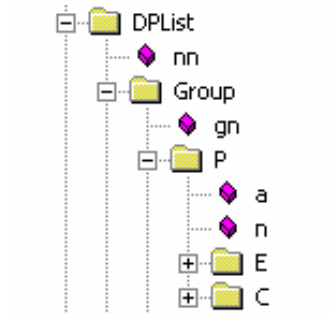
XML Processing Instruction	Type	Range [Unit]	Use	Def
Konfiguration Extern	Elem	-	req	1 ✔
Configuration Version	%s	[text]	opt	1 ✔

Die externe Konfigurationsdatei hat exakt die gleiche Struktur wie die ➲ lokale Konfigurationsdatei. Hier sind jedoch nur die Hauptelemente <**DPList**>, <**Connect**> oder <**Matrix**> zulässig. Diese können optional den Knotennamen **nn**=".." zur Identifikation führen (hellrot). Das Einfügen des Knotennamens erlaubt es, dass EINE externe Datei für verschiedene DxNode definiert werden kann. Andererseits kann die GLEICHE Parametrierung für mehrere DxNode verwendet werden, wenn der Knotenname fehlt.

Externe Datenpunktliste <DPList>

Element <**DPList**> definiert die Datenpunkte genau gleich wie für die ➲ lokale Konfiguration:

<**DxCnfExt**> Fortsetzung



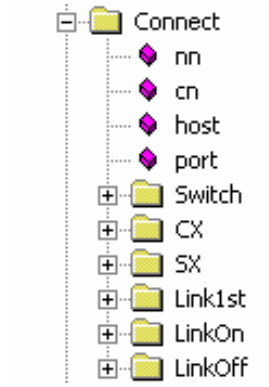
Konfiguration Extern	Type	Range [Unit]	Use	Def
Datenpunktliste	Elem	-	opt	x ✔
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Gruppe	Elem	-	req	x ✔
Group Name	%s	[A-Za-z0-9_]	req	1 ✔
Datenpunkt	Elem	-	req	x ✔
Address (application)	%s	[A-Za-z0-9_]	min1	1 ✔
Name (network)	%s	[text]	min1	1 ✔
Elementdaten	Elem	-	opt	1 ✔
Konditionierungsdaten	Elem	-	opt	1 ✔

Es können eine lokale und/oder mehrere externe <**DPList**> definiert werden. Die externe <**DPList**> ist mit dem Dateinamen **fn**=".." im Element <**DPList**> der lokalen Konfigurationsdatei zu referenzieren, damit sie beim Starten von DxNode automatisch eingefügt wird. Sie kann auch zur Laufzeit mit dem <**Include**> Befehl eingefügt werden. Dabei werden nur neue Datenpunkte angehängt, bestehende können zur Laufzeit nicht verändert oder gelöscht werden.

Externe Anschluss/Verbindung <Connect>

Element <**Connect**> definiert einen Anschluss bzw. eine Verbindung zu einem anderen Partner genau gleich wie für die ➲ lokale Konfiguration:

<**DxCnfExt**> Fortsetzung



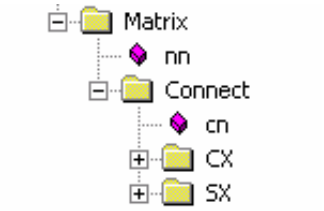
Konfiguration Extern	Type	Range [Unit]	Use	Def
Anschluss/Verbindung	Elem	-	opt	x ✔
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Connect Name (local)	%s	[A-Za-z0-9_]	req	1 ✔
Host Name or IP-Address	%s	[text]	opt	1ref
Port Addr or Service-Nr	%s	[text]	opt	1ref
Umschaltung/Vermittlung	Elem	-	opt	1 ✔
Client Matrix	Elem	-	opt	1 ✔
Server Matrix	Elem	-	opt	1 ✔
Verbindung 1st Start-Up	Elem	-	opt	1 ✔
Verbindung ON	Elem	-	opt	1 ✔
Verbindung OFF	Elem	-	opt	1 ✔

Es können mehrere lokale und/oder externe <**Connect**> definiert werden. Eine neue Verbindung wird jedoch nur beim Hochfahren d.h. nicht zur Laufzeit eingefügt. Die Verbindung kann daher nur mit dem Dateinamen **fn**="" im Element <**Connect**> der lokalen Konfigurationsdatei referenziert, jedoch nicht mit dem <**Include**> Befehl eingefügt werden.

Externe Kommunikationsmatrix <Matrix>

Element <**Matrix**> definiert die auszutauschenden Daten für eine oder mehrere Verbindungen genau gleich wie für die ➲ lokale Konfiguration:

<**DxCnfExt**> Fortsetzung



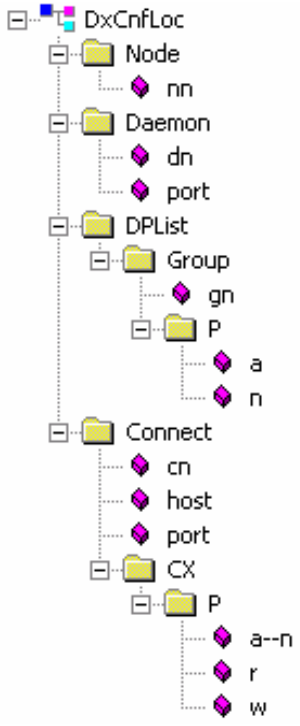
Konfiguration Extern	**Type**	**Range** [**Unit**]	**Use**	**Def**
Kommunikationsmatrix	Elem	-	opt	x ✔
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Verbindungsreferenz	Elem	-	opt	x ✔
Connect Name	%s	[A-Za-z0-9_]	req	1ref
Client Matrix	Elem	-	opt	1 ✔
Server Matrix	Elem	-	opt	1 ✔

Es können eine lokale und/oder mehrere externe <**Matrix**> definiert werden. Die extern definierte <**Matrix**> soll mit dem Dateinamen **fn**="" im Element <**Matrix**> in der lokalen Konfigurationsdatei referenziert werden, damit sie beim Hochfahren von DxNode automatisch eingefügt wird. Sie kann auch zur Laufzeit mit dem <**Include**> Befehl eingefügt werden und ersetzt dann die entsprechende aktuell laufende <**Matrix**> vollständig.

Beispiele für Lokale und Externe Konfiguration

Nur Lokale Konfiguration

Die nachfolgende Tabelle zeigt die erforderlichen Einträge einer lokalen Konfiguration für einen DxNode mit einer Datenpunktliste <**DPList**> und einem aktiven Anschluss <**Connect**> zum Rechner mit Name **host**="" und DxNode Daemon **port**="". Dabei ist der lokale Teilnehmer als Client <**CX**> der ausgewählten Datenpunkte <**P**> mit bi-direktionalem Datenaustausch d.h. mit Read **r**="" und Write **w**="" vom - bzw. zum Server definiert. Weitere Attribute und Elemente können gemäss den vorgängigen Kapiteln an den entsprechenden Stellen eingefügt werden.



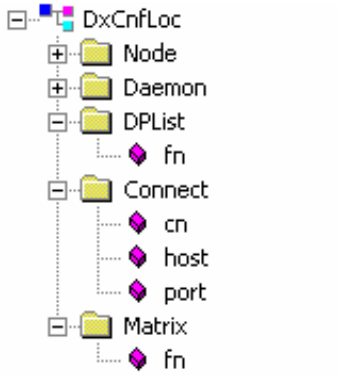
Konfiguration Lokal	Type	Range [Unit]	Use	Def
Grundeinstellungen	Elem	-	req	1
Node Name	%s	[A-Za-z0-9_]	req	1
Zugangsport	Elem	-	req	x
Daemon Name	%s	[A-Za-z0-9_]	req	1
Port Addr or Service-Nr	%s	[text]	req	ref
Datenpunktliste	Elem	-	req	1
Gruppe	Elem	-	req	x
Group Name	%s	[A-Za-z0-9_]	req	1
Datenpunkt	Elem	-	req	x
Address (application)	%s	[A-Za-z0-9_]	1req	1
Name (network)	%s	[text]	1req	1
Anschluss/Verbindung	Elem	-	opt	x
Connect Name	%s	[A-Za-z0-9_]	req	1
Host Name or IP-Address	%s	[text]	opt	1ref
Port Addr or Service-Nr	%s	[text]	opt	1ref
Client Matrix	Elem	-	opt	1
Datenpunkt-Auswahl	Elem	-	req	x
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
Read from Server Addr|Name	%s	[Addr|Name|wildcard|=]	req	1ref
Write to Server Addr|Name	%s	[Addr|Name|wildcard|=]	opt	1ref

Der Datenaustausch wird im obigen Beispiel direkt als so genannte Client Matrix <**CX**>, im Element <**Connect**> definiert. Das Element <**CX**>, auch Client Subskription genannt, kann auch im Element <**Matrix**> der derselben lokalen Konfigurationsdatei mit der entsprechenden Verbindungsreferenz <**Connect**> aufgeführt werden. In diesem Falle wird die oben definierte <**CX**> vollständig ersetzt d.h. sie soll weggelassen werden und es gilt nur die in der <**Matrix**> referenzierte Client Subskription.

☞ Die oben gezeigte Konfiguration ist die bevorzugte Darstellung, wenn keine separate <**Matrix**> gefordert ist. Das Element <**CX**> wird direkt am <**Connect**> angefügt.

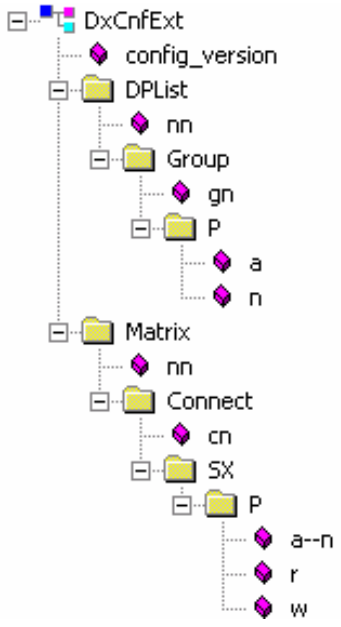
Lokale Konfiguration und Externe Referenz

Die nachfolgende Tabelle zeigt das gleiche Beispiel wie oben mit den erforderlichen Einträgen für die lokale Konfiguration und einer Referenz auf eine externe Konfiguration für <**DPList**> und <**Matrix**>. Der aktive Anschluss <**Connect**> zum Rechner mit Name **host**=".." und DxNode Daemon **port**=".." ist mit Ausnahme der zu übertragenden Daten fix parametriert. Die Datenpunktliste <**DPList**> und die <**Matrix**> haben nur den Eintrag **fn**=".." zur Referenzierung der externen Konfiguration, andere Attribute sind hier nicht zulässig. In allen anderen Elementen dürfen weitere Attribute und Elemente gemäss den nachfolgenden Kapiteln an den entsprechenden Stellen eingefügt werden.



Konfiguration Lokal	Type	Range [Unit]	Use	Def
Grundeinstellungen	Elem	-	req	1
Zugangsport	Elem	-	req	x
Datenpunktliste	Elem	-	req	1
File Name	%s	[FileName]	req	1ref
Anschluss/Verbindung	Elem	-	opt	x
Connect Name	%s	[A-Za-z0-9_]	req	1
Host Name or IP-Address	%s	[text]	opt	1ref
Port Addr or Service-Nr	%s	[text]	opt	1ref
Kommunikationsmatrix	Elem	-	opt	1
File Name	%s	[FileName]	req	1ref

In diesem Falle werden die Datenpunktliste <**DPList**> und die Kommunikationsmatrix <**Matrix**> beim Starten von DxNode aus der externen Konfigurationsdatei mit Namen **fn**=".." importiert. Die Methode erlaubt einen autarken Betrieb von DxNode, wenn die externe Konfigurationsdatei auf der lokalen Festplatte gespeichert ist. Die externe Konfigurationsdatei ist exakt gleich aufgebaut wie eine entsprechende lokale, mit einer Ausnahme: die Hauptelemente führen zusätzlich Knotennamen **nn**=".." zur Identifikation von DxNode. Dies ermöglicht, die <**DPList**> und <**Matrix**> für mehrere DxNode in einer Datei zu parametrieren - so könnte man z.B. nur EINE externe Konfigurationsdatei für ALLE DxNode definieren.

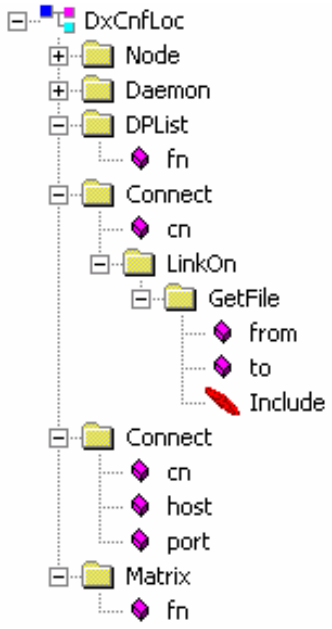


Konfiguration Extern	Type	Range [Unit]	Use	Def
Configuration Version	%s	[text]	opt	1
Datenpunktliste	Elem	-	opt	x
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Gruppe	Elem	-	req	x
Group Name	%s	[A-Za-z0-9_]	req	1
Datenpunkt	Elem	-	req	x
Address (application)	%s	[A-Za-z0-9_]	min1	1
Name (network)	%s	[text]	min1	1
Kommunikationsmatrix	Elem	-	opt	x
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Verbindungsreferenz	Elem	-	req	x
Connect Name	%s	[A-Za-z0-9_]	req	1ref
Client Matrix	Elem	-	opt	1
Datenpunkt-Auswahl	Elem	-	req	x
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
Read from Server Addr|Name	%s	[Addr|Name|wildcard|=]	req	1ref
Write to Server Addr|Name	%s	[Addr|Name|wildcard|=]	opt	1ref

Die externe Datei kann im Root Element eine Versionsbezeichnung **config_version**=".." führen. Diese dient der Identifikation der aktuell geladenen Konfiguration und kann über die ➲ internen Datenpunkte [Prefix][NodeName].value.**dp_version** sowie [Prefix][ConnectName].matrix.**cx_version** und [Prefix][ConnectName].matrix.**sx_version** für die entsprechenden Elemente abgefragt werden.

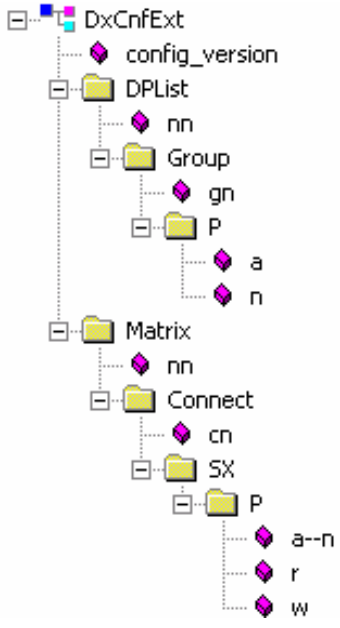
Lokale Konfiguration und Externe Referenz mit Download

Die folgende Tabelle zeigt das gleiche Beispiel wie oben. Der aktive Anschluss <**Connect**> dient hier jedoch zusätzlich dem Herunterladen (Download) der externen Konfiguration vom Partnerknoten. Dies geschieht mittels Steuerelement <**LinkOn**>, wenn die Verbindung (re)initialisiert und erfolgreich erstellt wurde. Dabei wird mit dem Befehl <**GetFile**> zuerst die externe Konfigurationsdatei auf die lokale Festplatte geladen, anschliessend wird sie mit dem Befehl <**Include**> zur Laufzeit eingefügt.



Konfiguration Lokal	Type	Range [Unit]	Use	Def
Grundeinstellungen	Elem	-	req	1
Zugangsport	Elem	-	req	x
Datenpunktliste	Elem	-	req	1
File Name	%s	[FileName]	req	1ref
Anschluss/Verbindung	Elem	-	opt	x
Connect Name	%s	[A-Za-z0-9_]	req	1
Steuerelement	Elem	-	opt	1
GetFile Instruktion	Elem	-	opt	x
Source File	%s	[FileName]	req	1ref
Target File	%s	[FileName]	req	1
Include File Instruktion	Elem	-	opt	1
Anschluss/Verbindung	Elem	-	opt	x
Connect Name	%s	[A-Za-z0-9_]	req	1
Host Name or IP-Address	%s	[text]	opt	1ref
Port Addr or Service-Nr	%s	[text]	opt	1ref
Kommunikationsmatrix	Elem	-	opt	1
File Name	%s	[FileName]	req	1ref

Die lokal gespeicherte Datei macht DxNode autark. Eine Initialisierung der Verbindung und damit ein Download kann jederzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].cmdio.state ausgelöst werden. Die <**Matrix**> ist hier für die gleiche Verbindung definiert d.h. der Partnerknoten hat Zugriff auf die externe Konfiguration und ist gleichzeitig Server für die Client Matrix <**CX**>. Eine ggf. veränderte Matrix bewirkt zwar eine erneute Initialisierung für den Datenabgleich, aber die externe Datei wird nicht mehr heruntergeladen, weil sie schon vorhanden ist. Natürlich kann man auch eine separate Verbindung, ausschliesslich zum Herunterladen der externen Konfiguration, parametrieren.



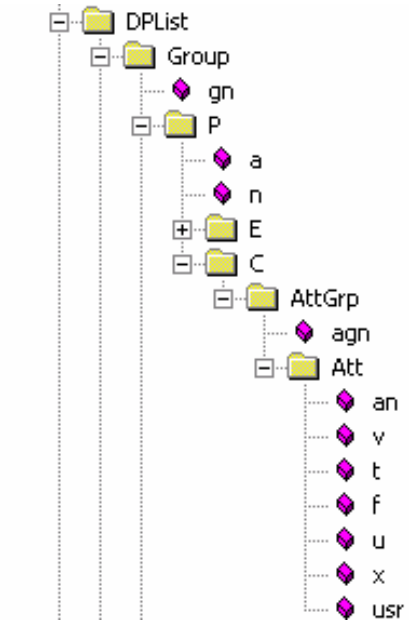
Konfiguration Extern	Type	Range [Unit]	Use	Def
Configuration Version	%s	[text]	opt	1
Datenpunktliste	Elem	-	opt	x
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Gruppe	Elem	-	req	x
Group Name	%s	[A-Za-z0-9_]	req	1
Datenpunkt	Elem	-	req	x
Address (application)	%s	[A-Za-z0-9_]	min1	1
Name (network)	%s	[text]	min1	1
Kommunikationsmatrix	Elem	-	opt	x
Node Name	%s	[A-Za-z0-9_]	opt	1ref
Verbindungsreferenz	Elem	-	req	x
Connect Name	%s	[A-Za-z0-9_]	req	1ref
Client Matrix	Elem	-	opt	1
Datenpunkt-Auswahl	Elem	-	req	x
Addr (appl.) | Name (network)	%s	[Addr|Name|wildcard]	xor	1ref
Read from Server Addr|Name	%s	[Addr|Name|wildcard|=]	req	1ref
Write to Server Addr|Name	%s	[Addr|Name|wildcard|=]	opt	1ref

Bemerkung: Die externe Konfigurationsdatei ist identisch mit derjenigen vom vorangegangenen Beispiel.

Proprietäre Konditionierungsdaten <C>

Die Datenpunktliste <**DPList**> verfügt pro Datenpunkt <**P**> optional über ein Element <**C**> das Konditionierungsdaten für die lokale Anbindung führen kann. Diese Daten werden zwar von DxNode ignoriert, können aber mit dem Download der <**DPList**> dem Anbinder zur Verfügung gestellt werden. Sie sind ggf. von der Anbindung direkt aus der XML-Datei zu auszulesen:

<**DxCnfLoc**>|<**DxCnfExt**> Fortsetzung



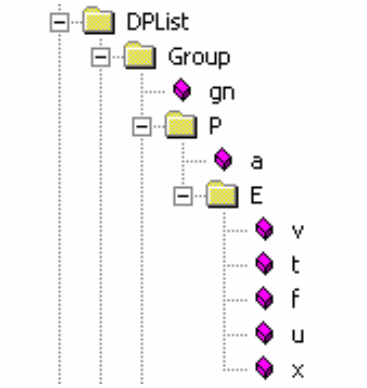
Konfiguration Lokal | Extern	Type	Range [Unit]	Use	Def
Datenpunktliste	Elem	-	opt	1
Gruppe	Elem	-	req	x
Group Name	%s	[A-Za-z0-9_]	req	1
Datenpunkt	Elem	-	req	x
Address (application)	%s	[A-Za-z0-9_]	min1	1
Name (network)	%s	[text]	min1	1
Elementdaten	Elem	-	opt	1
Konditionierungsdaten	Elem	-	opt	1
Attribute Group	Elem		req	x
Attribute Group Name	%s	[text]	req	1
Attribute	Elem	<![CDATA[..]]>	req	x
Attribute Name	%s	[text]	req	1
Value	%s	[text]	opt	1
Timestamp	%yh	[YMDhmsx]	opt	1
Format and Datatype	%s	[selection]	opt	1
Engineering Unit	%s	[selection]	opt	1
Text Information	%s	[text]	opt	1
User Name	%s	[text]	opt	1

Die Methode erlaubt eine beliebige Anzahl frei definierbarer Attribute zu parametrieren. Dabei ist der Attributname z.B. als [**agn**=".."]_[**an**=".."] definierbar. Die Parameterkonstanten kann als CDATA (Character Data) im Element <**Att**> oder im Attribut **v**=".." abgelegt werden. Der User Name **usr**=".." zeigt wer die letzte Wertänderung zu welcher Zeit **t**=".." vorgenommen hat.

☞ Die Konfigurationsdaten <**C**> werden von DxNode nicht verarbeitet sondern nur der Anbindung zur Verfügung gestellt.

Als **Alternative** kann man natürlich auch lokale Datenpunkte als Parameterkonstanten definieren:

<**DxCnfLoc**>|<**DxCnfExt**> Fortsetzung



Konfiguration Lokal | Extern	Type	Range [Unit]	Use	Def
Datenpunktliste	Elem	-	opt	1
Gruppe	Elem	-	req	x
Group Name	%s	[A-Za-z0-9_]	req	1
Datenpunkt	Elem	-	req	x
Address (application)	%s	[A-Za-z0-9_]	min1	1
Elementdaten	Elem	-	opt	1
Value	%s	[text]	opt	1
Timestamp	%yh	[YMDhmsx]	opt	1
Format and Datatype	%s	[selection]	opt	1
Engineering Unit	%s	[selection]	opt	1
Text Information	%s	[text]	opt	1

In diesem Fall werden nur lokale Adressen **a**=".." vergeben d.h. die Daten werden nur in der <**DPList**> aufgenommen, aber nicht übers Netz kommuniziert. Ein solcher "Parametrier-Datenpunkt" hat natürlich nur einen Bezug über die Adressierung zu (s)einem zugeordneten Datenpunkt. Der Vorteil dieser Variante ist aber, dass die Parameterkonstanten jederzeit online verändert und wie alle anderen Daten übertragen werden können.

Attribut Definition und Verwendung

Dieses Kapitel beschreibt alle Attribute in alphabetischer Reihenfolge. Die Beschreibung umfasst die Definition und Anwendung sowohl für die Konfiguration <**DxCnfLoc**> oder <**DxCnfExt**> wie auch für Telegramme <**X0**>. Damit wird einerseits die enge Beziehung zwischen der Parametrierung und deren Auswirkungen auf die Datenübermittlung dargestellt, andererseits dient die Beschreibung auch der Erläuterung des XML-Protokolls für den Datenaustausch. Folgende Funktionen zeigen z.B. diesen Zusammenhang:

- eine Subskription erfolgt durch Übermittlung der parametrierten Matrix <**CX**> oder <**SX**>
- ein Telegramm <**X0**> an das Partnersystem kann z.B. in <**LinkOn**> parametriert werden
- die Parametrierung von Standardwerten für Elementdaten <**P**><**E [..]**/> in der <**DPList**> oder in einem Steuerelement <**Link**..> definiert eigentlich ein internes Telegramm an DxNode

Für die folgenden Darstellungen gilt die Legende in Kapitel ➲ Darstellungskonventionen oben.

a="Address (application)" n="Name (network)"

- *Definition*

Die Lokale Adresse und/oder der Netzwerk Name der einzelnen Datenpunkte <**P**> werden in der <**DPList**> definiert und sind darstellbar als zwei gleichwertige Adressräume (Applikation und Netzwerk) im lokalen DxNode. Gültige Einträge sind:

```
<DPList>
   <Group gn="..">
      <P a="[a-zA-Z0-9_./]">[<E ... />]</P> nur lokale Adresse oder
      <P n="[text]">[<E ... />]</P> nur Netzwerk Name oder
      <P a="[a-zA-Z0-9_./]" n="[text]">[<E ... />]</P> Adresse und Name
```

Die lokale Adresse ist vorzugsweise IEC-1131 konform und erlaubt zusätzliche Trennzeichen um z.B. SPS-Adressen zu genügen. Datenpunkt Adresse und Name sind hier vollständig einzutragen und sollen nur sichtbare Zeichen enthalten. Generell sind in der <**DPList**> Wildcards (*?) nicht zulässig und XML-Steuerzeichen (<>'&") oder reservierte Zeichen (#;~^|) sind zu vermeiden. Die Einschränkungen sind im XML Schema festgelegt.

Standardwerte können als Attribute der Elementdaten <**E**> angefügt werden. Zum Beispiel bedeutet <**P a**="DB12W45" **n**="Valve21_pos"><**E u**="%" **v**="0" **q**="u" /></**P**>: Datenpunkt mit lokaler Adresse "DB12W45", Netzwerk Name "Valve21_pos", Einheit "%", Wert "0" und Qualität "u" (uncertain).

- *Anwendung*

Die Adresse oder der Name beschreiben für eine Subskription <**CX**> oder <**SX**> (Daueraufträge), für einen Single Request <**QX**> (Einzelabfrage), einen oder mehrere Datenpunkte <**P**> im Client. Die Angabe erfordert die korrespondierende Adresse bzw. den Namen **r**=".." und/oder **w**=".." im Server. Gültige Einträge sind:

entweder Adresse(n) ...

```
<CX> und/oder
<SX> oder
<QX> (nur in Telegramm)
   <P a="[Addr|Wildcard]"r="[Addr|Wildcard|=]" [w="[Addr|Wildcard|=]"][gn=".."][attr=".."][..]/>
```

oder Name(n) ...

```
<CX> und/oder
<SX> oder
<QX> (nur in Telegramm)
   <P n="[Name|Wildcard]"r="[Name|Wildcard|=]"
[w="[Name|Wildcard|=]"][gn=".."][attr=".."][..]/>
```

Adressen und Namen dürfen in einer Subskription nicht gemischt werden. Zur Selektion mehrerer Datenpunkte sollen Wildcards (*?) in **a**|**n**=".." und/oder die Gruppennamen **gn**=".." der <**DPList**> verwendet werden. Die Anzahl der Wildcards in **a**|**n**=".." und **r**=".." bzw. **w**=".." muss übereinstimmen. Mit **attr**=".." können die zu übertragenden Attribute im Telegram definiert werden.

Der Ausdruck <**P n**="*io*" **r**="*rd*" **w**="*wr*" /> bedeutet, dass z.B. ein Datenpunkt im Client mit Name "Temp_io.val" unter dem Namen "Temp_rd.val" vom Server gelesen - und mit Name "Temp_wr.val" an den Server übermittelt würde.

Die Adresse oder der Name beschreibt in einem Steuerelement <**Link**...> den Zieldatenpunkt <**P**> zum einstellen bzw. übermitteln von Elementdaten <**D**|**E**|**e** ... />. Gültige Einträge sind:

<**Link1st**> und/oder
<**LinkOn**> und/oder
<**LinkOff**>
  <**P a**="[Addr|Wildcard]"><**D**|**E**|**e** [**v**=".."][..]/></**P**> und/oder
  <**P n**="[Name|Wildcard]"><**D**|**E**|**e** [**v**=".."][..]/></**P**>

Die lokale Adresse bzw. der Netzwerk Name dürfen auch Wildcards (*?) enthalten, der entsprechende Wert **v**=".." etc. wird dann an alle zutreffenden Datenpunkte in DxNode übermittelt. Z.B. bedeutet der Ausdruck <**P a**="*.val"><**E v**="0" /></**P**>, dass alle Datenpunkte deren Adresse mit ".val" enden, bei Ausführung des entsprechenden Steuerelementes auf "0" gesetzt werden.

Lokale Adresse oder Netzwerk Name beschreiben in Telegrammen <**X0**> den Zieldatenpunkt <**P**> im Client beim lesen <**D**|**E**> oder denjenigen im Server beim schreiben <**e**>. Gültige Darstellungen sind:

<**X0**>[<**CXR**|**SXR**|**QXR**|**HXR**>]
  <**P a**="[Addr|Wildcard]"><**D**|**E**|**e** [**n**="SourceName"][**a**="SourceAddr"][**v**=".."][..]/></**P**> und/oder
  <**P n**="[Name|Wildcard]"><**D**|**E**|**e** [**n**="SourceName"][**a**="SourceAddr"][**v**=".."][..]/></**P**>

Die lokale Adresse bzw. der Netzwerk Name dürfen zur Beschreibung der Zieldatenpunkte <**P**> auch Wildcards (*?) enthalten, der entsprechende Wert **v**=".." etc. wird dann an alle treffenden Datenpunkte im Zielknoten übermittelt.

Die Adresse und/oder der Name können nach Vorgabe durch **attr**="+a+n" in <**CX**> und/oder <**SX**> oder <**QX**> zusätzlich in den Elementdaten <**D**|**E**|**e** ... /> enthalten sein. Adresse und/oder Name bezeichnen in <**D**|**E**|**e**> immer den Quellendatenpunkt des Senders und werden nur dann übermittelt, wenn sie sich von der Adresse bzw. vom Namen des Zieldatenpunktes unterscheiden, zum Beispiel:

<**P a**="ZielAdresse"><**E n**="QuellenName" **v**=".."/></**P**> → ZielAdresse=QuellenAdresse
<**P n**="ZielName"><**E n**="QuellenName" **a**="QuellenAdresse" **v**=".."/></**P**> → ZielName≠QuellenName

alive="Alive Timeout"

- *Definition und Anwendung*

Die Lebenszeichen-Überwachungszeit in [sec] kann global für alle Verbindungen im Element <**Node**> - oder individuell pro Verbindung im Element <**Connect**> eingestellt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **alive**="[1..9999]" [..]/>

und/oder ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **alive**="[1..9999]" [..]>

Der Standardwert ist auf **alive**="30" [sec] eingestellt, dies bedeutet, dass alle 15 sec ein Telegramm <**X0 sid**=".." **rid**=".." **t**=".."><**Alive**/></**X0**> übermittelt wird, wenn keine anderen Daten übermittelt werden. Der Partner Knoten antwortet dann ggf. mit <**X0 sid**=".." **rid**=".." **t**=".."><**AliveR**/></**X0**>.

Der Standardwert wird durch den Eintrag in <**Node**> ersetzt. Wenn ein Eintrag in <**Connect**> vorhanden ist, dominiert dieser. Zum Beispiel bedeutet <**Connect cn**="Test" **alive**="5">, dass über diese Verbindung mindestens alle 2.5 sec ein Telegramm übermittelt wird.

Die Überwachungszeit kann für jede Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].cmdio.**alive** abgefragt oder verändert werden. Die parametrierte Vorgabe **alive**=".." wird dabei nicht verändert.

attr="Attributes"

- *Definition und Anwendung*

Der Attribut-Parameter bestimmt, welche Daten in <**D**|**E**|**e** ... /> wann mit <**X0**><**P**> Telegrammen übermittelt werden. Mit dem Eintrag wird auch festgelegt, ob die Werte gespeichert und weitergeleitet werden sollen (Store&Fwd). Der Parameter kann pro Subskription <**CX**>, <**SX**>, Abfrage <**QX**> und/oder deren untergeordneter Datenpunkt-Auswahl <**P**> definiert werden. Gültige Einträge sind:

<**CX attr**="[S,a,n,v,t,q,f,s,i,u,x,c]" [..]> und/oder
<**SX attr**="[S,a,n,v,t,q,f,s,i,u,x,c]" [..]> oder
<**QX attr**="[S,a,n,v,t,q,f,s,i,u,x,c]" [..]> (nur in Telegramm)
  <**P a**=".." **r**=".." [**w**=".."] **attr**="[S,a,n,v,t,q,f,s,i,u,x,c]" [..]/> oder
  <**P n**=".." **r**=".." [**w**=".."] **attr**="[S,a,n,v,t,q,f,s,i,u,x,c]" [..]/>

... wobei der Ausdruck [S,a,n,v,t,q,f,s,i,u,x,c] die Liste der zu übertragenden Elementwerte darstellt.

Das **Trennzeichen** (anstelle des Kommas) bestimmt, wann ein Wert übertragen wird: (**.**)="nur bei Änderung und zur Initialisierung", (**+**)="immer wenn vorhanden", (**-**)="nie, keine Übertragung" d.h.:

der Eintrag ...	führt in Telegrammen <**X0**><**P**> zur Übertragung von Elementdaten ...
S	<**E** [Store&Fwd der definierten Elementdaten]/>, nur gültig für <**SX**> bzw. <**SX**><**P**>
+a	<**D**|**E**|**e a**="[SourceAddr]" />, wenn ungleich <**P a**="[Addr]"> d.h. bei Umbenennung
+n	<**D**|**E**|**e n**="[SourceName]" />, wenn ungleich <**P n**="[Name]"> d.h. bei Umbenennung
.v oder +v	<**D**|**E**|**e v**="[Value]" />, wenn vorhanden
.t oder +t	<**D**|**E**|**e t**="[Timestamp]" [**tc**="TimeCorrective"]/> ** mit Zeitkorrektur nur wenn **tc**="[NonZero]"
.q oder +q	<**D**|**E**|**e q**="[Quality]" />, wenn vorhanden
.f oder +f	<**D**|**E**|**e f**="[Format" />, wenn vorhanden
.s oder +s	<**D**|**E**|**e s**="[Status]" />, wenn vorhanden
.i oder +i	<**D**|**E**|**e i**="[Index]" />, wenn vorhanden
.u oder +u	<**D**|**E**|**e u**="[Unit]" />, wenn vorhanden
.x oder +x	<**D**|**E**|**e x**="[Text]" />, wenn vorhanden
.c oder +c	<**D**|**E**|**e c**="[EventCounter]" />

Die **Standardeinstellung** ist **attr**="-a-n+v+t+q.f.s.i.u.x-c" d.h. eine mögliche Umbenennung von Name oder Adresse sowie der Ereigniszähler werden in <**D**|**E**|**e**> nicht angezeigt und mit Ausnahme von Value, Timestamp und Quality werden die Elementdaten nur bei Änderung oder zur Initialisierung übertragen. Ein entsprechendes Telegramm würde so aussehen:

<**X0**><**P n**|**a**="..."><**D**|**E**|**e v**=".." **t**=".." [**tc**=".."] **q**=".." [**f**="..."][**s**="..."][**i**="..."][**u**="..."][**x**="..."]/>
</**P**></**X0**>

Eine allfällige Zeitstempelkorrektur **tc**="[NonZero]" wird automatisch zusammen mit **t**=".." eingefügt.

Die Standardeinstellung wird durch den Eintrag nur modifiziert (nicht ersetzt). Wenn ein Eintrag in <**SX**> und <**P**> vorhanden ist, gelten beide Einträge wobei ggf. der in <**P**> dominiert. Zum Beispiel definiert die Eingabe <**SX attr**="+f"> einen Server mit kontinuierlicher Übermittlung des Formates **f**="..", zusammen mit der Eingabe <**P n**="X*" **r**="=" **attr**="S" /> bedeutet dies, dass zusätzlich alle Datenpunkte deren Namen mit "X" beginnen, im Store&Fwd Buffer gespeichert werden. Die Eingabe <**CX**><**P n**="*Test" **r**="=" **attr**="+s+u" /></**CX**> definiert einen Client für Datenpunkte mit Endung "Test" im Namen und kontinuierlicher Anzeige des Status mit Masseinheit (z.B. für DxMonitor).

Die **Adresse** und/oder der **Name** können nach Vorgabe durch **attr**="+a+n" zusätzlich in <**D**|**E**|**e** ... /> enthalten sein. Die Adresse oder der Name bezeichnen in <**D**|**E**|**e**> immer den Quellendatenpunkt des Senders und werden nur übermittelt, wenn sie sich von der Adresse bzw. vom Namen des Zieldatenpunktes <**P a**|**n**="..."> unterscheiden, mit **attr**="+a+n" bedeutet deshalb zum Beispiel ...

<**P a**="ZielAdresse"><**E n**="QuellenName" **v**=".."/></**P**> → ZielAdresse=QuellenAdresse
<**P n**="ZielName"><**E n**="QuellenName" **a**="QuellenAdresse" **v**=".."/></**P**> → ZielName≠QuellenName

c="Event Counter" (reserved)

- *Definition und Anwendung*

Ein Ereigniszähler wird pro Datenpunkt <**P**> mitgeführt. Der Zähler wird beim Start von DxNode auf Null gestellt und kann zur Laufzeit nur abgefragt werden (Read Only). Gültige Darstellungen sind:

<**X0**><**P**><**E** [**v**=".."] **c**="[0...2147483647]" [..]/></**P**></**X0**>

Der Standardwert ist **c**="0". Der Zähler kann z.B. von einer Anwendung oder vom DxMonitor abgefragt werden um die Anzahl Änderungen eines Datenpunktes seit Knotenstart anzuzeigen.

cn="Connect Name"

- *Definition*

Für parametrierte Verbindungen ist ein eindeutiger Name im Element <**Connect**> zu definieren. Nur diese, mit einem Namen definierten Verbindungen können von anderen Teilnehmern referenziert oder zur Laufzeit über die <**Matrix**> parametriert werden. Gültige Einträge für einseitig parametrierte Verbindungen im aktiven Knoten sind:

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**="[A-Za-z0-9_]" **host**="[Host/IP]" **port**="[DaemonPort]" [..]>
    <**CX**> und/oder <**SX**>

In diesem Falle ist im Partnerknoten keine weitere Parametrierung erforderlich. Der Datenaustausch wird ausschliesslich lokal definiert und vom Partner **host**=".." mit Daemon **port**=".." entsprechend ausgeführt. Im Partnerknoten sind jedoch KEINE internen Datenpunkte für diese Verbindung verfügbar. Wenn dies gewünscht ist, dann muss die Verbindung beidseitig parametriert werden.

Für beidseitig parametrierte Verbindungen sind in beiden Partnerknoten vorzugsweise dieselben Namen zu definieren. Für beidseitig parametrierte redundante Verbindungen MUSS der Verbindungsname in den beiden redundanten Rechnern gleich lauten. Gültige Einträge sind für den aktiven Knoten ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**="[A-Za-z0-9_]" **host**="[Host/IP]" **port**="[DaemonPort]" [..]>
    <**Switch** [**cn**=".."]/>
    [<**CX**> und/oder <**SX**>]

und den passiven Knoten (oben aufgerufen mit **host**=".." und **port**="..") ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**="[A-Za-z0-9_]" [..]>
    [<**CX**> und/oder <**SX**>]

In diesem Falle werden in beiden Partnerknoten für dieselbe Verbindung interne Datenpunkte erzeugt. Dabei kann die Subskription <**CX**> und/oder <**SX**> entweder im aktiven oder im passiven Knoten parametriert werden. Die Verbindung wird dann vom passiven Knoten mittels Befehl <**Switch**/> (switch to named connect) an die gewünschte Verbindung weitergeleitet, wobei mit <**Switch cn**="../> auch ein anderer Name im passiven Knoten angesprochen werden kann.

In allen Fällen können die Steuerelemente <**Link1st**>,<**LinkOn**>,<**LinkOff**> angefügt - und damit vom Verbindungszustand abhängige Aktionen ausgelöst werden.

Der Verbindungsname ist vorzugsweise IEC-1131 konform da er zur Bezeichnung von internen Datenpunkten verwendet wird, die Einschränkungen sind im XML Schema festgelegt.

- *Anwendung*

Der Verbindungsname wird in der <**Matrix**> referenziert, wobei alle festen Vorgaben wie **host**="..", **port**=".." etc. sowie das Verhalten bei Verbindungsaufbau/-abbau erhalten bleiben. Mit der <**Matrix**> ist also nur die Subskription <**CX**> und/oder <**SX**>, auch Kommunikationsmatrix genannt, zu definieren und bei Bedarf zu übermitteln. Gültige Einträge sind:

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Matrix**>
    <**Connect cn**="[ConnectName]">

Die Verbindung mit diesem Verbindungsnamen muss im lokalen DxNode definiert/vorhanden sein.

Der Verbindungsname wird im Telegramm <**X0**> zur Identifikation bzw. zur Vermittlung an eine im Partnersystem vorparametrierte Verbindung verwendet. Gültige Darstellungen sind:

<**X0**>
  <**Connect cn**="[ConnectName]"><**Switch** [**tgt**=".."]/></**Connect**>

Mit dem Befehl <**Switch**/> wird die Vermittlung mit der vorparametrierten Verbindung [ConnectName] im Zielknoten angefordert. Fehlt das Element <**Switch**/>, dann wird nur der vorgängig angerufene Daemon-Prozess aufgerufen, d.h. es wird keine vorparametrierte Verbindung im Zielsystem angesprochen.

Hinweis→ nicht parametrierte Verbindungen haben keine ➲ internen Datenpunkte. Das Element <**Switch**/> ist im aufrufenden System zu parametrieren (siehe oben). Ein Verbindungsname in der Parametrierung <**Switch cn**="../> überschreibt den Namen <**Connect cn**="..> im aufrufenden Telegramm. Dies ist z.B. notwendig, wenn eine lokale Verbindung zum gleichen DxNode hergestellt wird. DxNode bestätigt die Vermittlung mit <**ConnectR cn**=".." /> und erzeugt automatisch den prozeduralen Ablauf für den Datenaustausch gemäss der vorgegebenen Parametrierung.

Der Verbindungsname [ConnectName] wird zur Bezeichnung ➲ interner Datenpunkte verwendet, dabei kann mit <**Node prefix**=".." [..]/> eine linksbündige Erweiterung eingefügt werden:

Lokale Adresse a=".."	**Beschreibung**
[Prefix][ConnectName].cmdio.sss	Befehls-Ein/Ausgabe (Bi-direktional Read/Write)
[Prefix][ConnectName].value.sss	Wert-Ausgabe/Anzeige (Read Only)
[Prefix][ConnectName].perf.sss	Performance-Anzeige (Read Only)
[Prefix][ConnectName].matrix.sss	Matrix-Information (Read Only oder Bi-direktional)

... wobei sss = Sub-Element-Bezeichnung = Attribut-Name bei parametrierbaren Items,
(Performance-Werte werden periodisch gemäss **perf_cycle**=".." updated/ausgegeben)

Der Verbindungsname [ConnectName] bezeichnet auch die ggf. für den Server anzulegende interne Datei für Store&Fwd Meldungen. Letztere werden mit **attr**="S..." für die Subskription <**SX**> im Server definiert und können dann von Clients gelesen und nachgeführt werden.

cn_list="Connect List" (DxMonitor)

- *Definition und Anwendung*

DxMonitor erlaubt das Überwachen und Aufzeichnen mehrerer Verbindungen <**Connect**> in DxNode. Hierzu wird eine Liste von **cn**="[ConnectName]" an DxNode als Befehl:

<**X0**><**Connect cn_list**="[aa;bb;cc]" [**verbose**=".."] [..]/>[<**CX**>|<**SX**>]</**X0**> (Befehl an DxNode)
<**Y0**><**Connect cn_list**="[aa;bb;cc]" [**verbose**=".."] [..]/>[<**CX**>|<**SX**>]</**Y0**> (DxLog Datei)

In der Log Datei wird der Befehl als <**Y0**> Telegramm ausgegeben. Zulässiger Wert [aa;bb;cc] ist eine Liste von Connect Namen, getrennt durch Semicolon (;).

☞ Attribut **cn_list**=".." kann nicht konfiguriert werden,
es kommt daher in den Konfigurationsdateien nicht vor.

config_level="Configuration Level"

- *Definition und Anwendung*

Parameter zur Freigabe/Sperrung automatischer Konfiguration von Datenpunkten über Verbindungen. Der Parameter kann global für alle Verbindungen im Element <**Node**> - oder individuell pro Verbindung im Element <**Connect**> eingestellt werden. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." config_level="[0|1]" [..]/>
```

und/oder ...

```
<DxCnfLoc> oder <DxCnfExt>
   <Connect cn=".." config_level="[0|1]" [..]/>
```

Der Standardwert ist **config_level**="0", dies bedeutet, dass DxNode keine neuen Datenpunkte <**P**> erzeugt d.h. für nicht definierte Datenpunkte in der <**DPList**> folgende Meldung in die Log-Datei schreibt: <**E2 t**=".." **cn**=".." **msg**="[2010] DP not found: ..."></**E2**>.

Mit **config_level**="1" werden neue Datenpunkte zur Laufzeit automatisch erzeugt sofern sie nicht bereits in der <**DPList**> definiert sind. In diesem Fall, wird folgende Meldung in die Log-Datei geschrieben: <**E4 t**=".." **cn**=".." **msg**="[4011] Insert Item: ..."></**E4**>.

☞ Automatisch erzeugte Datenpunkte <**P**> müssen - wenn erforderlich - beide Adressräume führen d.h. mit Telegrammen <**X0**><**P a**=".." **n**=".."/></**X0**> generiert werden weil für einen einmal erzeugten Datenpunkt die Adressen nicht mehr geändert werden können. Automatisch erzeugte Datenpunkte werden nicht in die <**DPList**> der Konfigurationsdatei geschrieben, d.h. bei Neustart von DxNode müssen diese Datenpunkte zur Laufzeit wieder erzeugt werden.

config_version="Configuration Version"

- *Definition und Anwendung*

Alle lokalen und externen Konfigurationsdateien können im Root Element <**DxCnfLoc**> bzw. <**DxCnfExt**> eine individuelle Konfigurationsversion führen. Gültige Einträge sind:

```
<?xml version="1.0" encoding=".."?>
<DxCnfLoc config_version="[anyText]"> oder
<DxCnfExt config_version="[anyText]">
```

Die Versionsbezeichnung dient der Versionisierung d.h. zur Identifikation der aktuellen Konfiguration. Sie wird der <**DPList**> sowie den Subskriptionen <**CX**> bzw. <**SX**> der Verbindungen <**Connect**> zugewiesen, wobei die Version der lokalen Konfiguration den lokal definierten Elementen - und die Version der externen Konfiguration den extern definierten Elementen zugeordnet wird.

Die Version kann für die entsprechenden Elemente zur Laufzeit über die ➲ internen Datenpunkte [Prefix][NodeName].value.**dp_version** sowie [Prefix][ConnectName].matrix.**cx_version** bzw. [Prefix][ConnectName].matrix.**sx_version** abgefragt werden.

DxNode liefert also für <**DPList**>, <**CX**> und <**SX**> immer die tatsächliche Konfigurationsversion, unabhängig davon ob sie lokal oder extern definiert wurde. Andererseits übernimmt DxNode die definierten Versionsbezeichnungen ohne weitere Kontrolle d.h. bei grösseren Netzwerken ist ein entsprechendes Versionsmanagement sinnvoll.

dn="Daemon Name"

- *Definition*

Für den Verbindungsaufbau ist immer zuerst ein so genannter Daemon im Partnersystem anzurufen, für den ein eindeutiger Name im Element <**Daemon**> zu definieren ist. Der Daemon unterstützt sowohl nicht-parametrierte Verbindungen (z.B. für DxMonitor oder nur im Partnersystem definierte) wie auch parametrierte, auf welche nach der Verbindungsherstellung umgeschaltet werden kann. Der Daemon stellt den einzigen Zugangspunkt an DxNode dar (es sind jedoch mehrere Daemone parametrierbar), er unterstützt ein "Listener Port" (Horcher) über das sich alle Teilnehmer anmelden müssen. Mit dem Zugriffsschutz des normalerweise einzigen Listener Ports, kann DxNode auf einfache Weise vor unerwünschtem Zugang geschützt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Daemon dn**="[A-Za-z0-9_]" **port**=".." [..]/>

Der Daemon-Name ist vorzugsweise IEC-1131 konform da er zur Bezeichnung von internen Datenpunkten verwendet wird, die Einschränkungen sind im XML Schema festgelegt.

- *Anwendung*

Der Daemon-Name [DaemonName] wird zur Bezeichnung ➲ interner Datenpunkte verwendet, dabei kann mit <**Node prefix**=".." [..]/> eine linksbündige Erweiterung eingefügt werden:

Lokale Adresse a=".."	**Beschreibung**
[Prefix][DaemonName].cmdio.sss	Befehls-Ein/Ausgabe (Bi-direktional Read/Write)
[Prefix][DaemonName].value.sss	Wert-Ausgabe/Anzeige (Read Only)

... wobei sss = Sub-Element-Bezeichnung = Attribut-Name bei parametrierbaren Items

encoding="XML Encoding"

- *Definition und Anwendung*

Die Konfigurationsdateien sollen mit einer so genannten Processing Instruction <?**xml**?> die Version des XML Standards und den verwendeten Zeichensatz (Encoding) mitführen. Gültige Einträge sind:

<?**xml version**="1.0" **encoding**="[Encoding]"?>
<**DxCnfLoc**> oder <**DxCnfExt**>

Die XML Standardwerte für die Processing Instruction sind **version**="1.0" und **encoding**="UTF-8" gemäss http://www.unicode.org, dies bedeutet, dass die Datei dem "XML Standard Version 1.0" entspricht und die Daten mit dem Zeichensatz "Unicode Transformation Format 8-Bit" dargestellt sind. Für Westeuropa ist jedoch **encoding**="ISO-8859-1" (ISO Latin-1) sinnvoll, weil die meisten Anbindungen UTF-8 nicht unterstützen und damit alle String-Variablen umwandeln müssten.

Eigentlich sollte auch jedes Telegramm mit dem "Overhead" <?**xml version**="1.0" **encoding**=".."?> beginnen, um einem Anwenderprogramm anzuzeigen, welcher Standard verwendet wird, ➲ siehe auch http://www.w3.org/XML. Für DxNode sind diese Standards jedoch netzweit identisch festzulegen. Damit können die Telegramme ohne "Overhead" schlank gehalten werden um einen hohen Datendurchsatz zu erreichen und auch simple Anbindungen (z.B. SPS) zu erlauben. Das ist deshalb möglich, weil DxNode sprachunabhängig ist d.h. die Encoding Instruction nicht verwendet und damit nur die XML-Syntax und definierten Inhalte der XML-Telegramme überprüfen muss.

☞ Das Encoding soll netzwerkweit für alle Konfigurationsdateien einheitlich sein, sonst werden Umlaute bzw. ASCII Zeichen grösser #x7F ggf. unterschiedlich interpretiert. Ohne Encoding Instruction erzeugt z.B. der XML Notepad für die Zeichen **ä, ö, ü** die UTF-8 Codes **ä**=#xC3A4, **ö**=#xC3B6, **ü**=#xC3BC, die dann mit dem ISO Latin-1 Zeichensatz als zwei Zeichen **Ã¤, Ã¶, Ã¼** dargestellt werden.

Unabhängig vom Encoding sind zur Darstellung von Elementdaten nur die ASCII-Zeichen #x9 [TAB], #xA [$^{L}_{F}$], #xD [$^{C}_{R}$] und #x20..#x7F sowie alle weiteren Zeichen des netzweit definierten Zeichensatzes zulässig. Zusätzlich sind die folgenden ASCII-Zeichen gemäss http://www.w3.org/TR/xmlschema-2 für XML reserviert oder sollten ggf. für die entsprechende Bedeutung in DxNode reserviert bleiben:

HEX- ASCII-Zeichen	**XML**	**Reservierte XML Entities**	
#x22 ["]	&quot;	Anführungszeichen (quotation mark)	Diese Zeichen sind in Textwerten (strings) durch die vorgegebenen XML-Entities (Escape-Sequenzen), dargestellt als &...; zu ersetzen
#x26 [**&**]	&amp;	Und-Zeichen (ampersand)	
#x27 [']	&apos;	Anführungsstrich (apostroph)	
#x3C [**<**]	&lt;	Kleiner-Zeichen (less than)	
#x3E [**>**]	&gt;	Grösser-Zeichen (greater than)	
HEX- ASCII-Zeichen	**XML**	**Wildcard Zeichen**	
#x2A [*****]	*	Sternchen (asterisk="any number of any char")	Diese Zeichen definieren Treffer in Masken/Ausdrücken
#x3F [**?**]	?	Fragezeichen (questionmark="any char")	
HEX- ASCII-Zeichen	**XML**	**Reservierte Sonderzeichen**	
#x23 [**#**]	#	Kreuzzeichen in XML Entity (sharp/number sign)	Diese Zeichen sind für spezielle Darstellungen reserviert und sind entsprechend anzuwenden
#x24 [**$**]	$	Dollarzeichen für Markierung (dollar sign)	
#x3B [**;**]	;	Strichpunkt für Auflistung (semicolon)	
#x20#x7C#x20 [**|**]	|	Array Delimiter in **v**=".." (space+bar+space)	

f="Format and Datatype" siehe auch v=".."

- *Definition und Anwendung*

Format und Datentyp werden mit einem C Format Specifier "printf" dargestellt und beziehen sich auf den Wert **v**="..". Sie sind bei Bedarf in der <**DPList**> oder in den Steuerelementen <**Link**..> pro Datenpunkt-Auswahl <**P**> als Elementdaten <**E**|**e f**="%.." /> definierbar. Gültige Einträge sind:

<**E**|**e** [**v**=".."] **f**="[%s|%X|%b|%f|%E|%L|%y|%h|%yh]" [..]/>

Der Standardwert ist **f**="%s". Datentypen sind gemäss http://www.w3.org/TR/xmlschema-2 normiert (Auswahl) und werden mit dem als Format %x.yz (z.Z. ohne flags, width oder precision) dargestellt. Dies bietet die Möglichkeit, die Werte **v**=".." später ggf. beliebig zu formatieren. Zurzeit sind folgende Formate (enumerated) zulässig:

Format	**Datentyp**	**Darstellung** (Default **v**="[Nullstring]")	**Zulässiger Bereich**
f="%s"	**string**	**v**="**aBcD**" Zeichen in XML	Beliebige Anzahl Zeichen ISO-8859-1/UTF-8
f="%X"	**hexBinary**	**v**="**0A7F**" Char in Hexadezimal	Beliebige Anzahl Char-Paare Hexadezimal
f="%b"	**boolean**	**v**="**1**, **0**" oder **true**, **false**	1/0 oder true/false
f="%f" **f**="%E"	**double**	**v**="**±1234.567**" Float / Real Value **v**="**±0.12E±34**" Exp. Darstellung	1 bis 15 Ziffern + Vorzeichen + Punkt ±1.7E+308 IEEE Double
f="%i"	**int**	**v**="**±123456**" Ganzzahl / Integer	-2147483648...+2147483647
f="%L"	**enumerated**	**v**="**123**" Stufenwert / Enum Level	-32768...+32767 (short integer)
f="%y"	**date**	**v**="**YYYY-MM-DD**" Jahr-Monat-Tag	1970-01-01...2037-12-31
f="%h"	**time**	**v**="**hh:mm:ss.xxx**" Std:min:sec.ms	00:00:00.000...23:59:59.999
f="%yh"	**dateTime**	**v**="**YYYY-MM-DDThh:mm:ss.xxx**" Telegramm <**X0**> ggf. mit [±hh:mm]	1970-01-01T00:00:00.000 [±hh:mm]... 2037-12-31T23:59:59.999 [±hh:mm]

Strings dürfen nur die die ASCII-Zeichen #x9 [TAB], #xA [$^{L}_{F}$], #xD [$^{C}_{R}$] und #x20..#x7F sowie alle weiteren Zeichen des netzweit gültigen Zeichensatzes enthalten, siehe ➲ **encoding**="..". Dabei sind die XML reservierten Zeichen [<, >, &, ", '] in XML-Entities (Escape Sequenzen) zu wandeln.

Zahlenwerte sollen gemäss W3C-Organisation mit Grossbuchstaben ohne (+) Zeichen dargestellt werden. Wenngleich DxNode alle Zahlenwerte verarbeitet (z.B. auch %x oder %e), so soll die Empfehlung von W3C eingehalten werden, um gängige Wandlungen zu unterstützen.

Datum und Zeitangaben sind immer Universal Time Coordinated (UTC), d.h. weltweit eindeutig, ohne Zeitzonenverschiebung oder Sommer/Winterzeitumschaltung. Zeitangaben sind nach ISO-8601 in [ms] formatiert, die Zeitzone [±hh:mm] kann ggf. im Zeitstempel von <**X0**> mitgeführt werden.

Enumerated Values sind normalerweise vom Typ "string" wie z.B. **q**=".." oder **u**="..". Das Format **f**="%L" ist vom Typ "short" und zeigt einen Stufenwert, der anwendungsseitig definiert ist. Es ermöglicht eine spätere Anzeige mit klassifizierten Texten z.B. "Aus", "Stufe1" etc. in DxNode.

Arrays haben ein Format (homogen), sie sind nur durch den Array Delimiter " | " im Wert **v**=".." z.B. als <**E v**="Wert1 | Wert2 | Wert3 | WertX"/> und nicht durch das Format **f**="%.." erkennbar. Dabei ist die ASCII-Zeichenfolge " | " (#x20#x7C#x20) als Array Delimiter definiert, diese Zeichenfolge ist daher in Array of Strings nicht zulässig.

fast_update="Control Flag" siehe upd_delay=".."

- *Definition und Anwendung*

Ein Flag zur verkürzten Transitionsanzeige kann im Element <**Node**> für alle Verbindungen eingestellt werden. Es dient dazu die Update-Verzögerung im Client minimal einzustellen. Für gültige Eingaben ➲ siehe **upd_delay**="Update Delay".

flush_cycle="Store&Fwd Flush Cycle"

- *Definition und Anwendung*

Der Auslagerungszyklus in [ms] für Store&Fwd Daten kann global für alle Verbindungen im Element <**Node**> eingestellt werden. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." flush_cycle="[500..1000000]" [..]/>
```

Der Standardwert ist **flush_cycle**="2000" [ms] und bedeutet, dass die mit **attr**="S" definierten Store&Fwd Daten <**P**> vom sendenden DxNode alle 2 sec auf die lokale Harddisk ausgelagert werden. Im empfangenden DxNode wird jeweils der Zeitstempel **t**=".." des zuletzt empfangenen Telegrammes <**X0**> gespeichert. Zur maximalen Sicherung von Ereignissen sind jedoch auch für Store&Fwd Daten redundante DxNode und Systeme empfohlen.

fn="File Name"

- *Definition und Anwendung*

Für die Elemente <**DPList**>, <**Connect**>, <**Matrix**> und <**Include**> kann eine separate bzw. externe Konfigurationsdatei referenziert werden, um dort parametrierbare Elemente zu definieren bzw. zu ersetzen. Dabei muss der Inhalt identisch mit einer vergleichbaren lokalen Konfiguration sein.

Die externe Datei kann eine Konfigurationsversion **config_version**=".." zur Identifizierung der Daten enthalten. Sie muss alle Elemente enthalten, die ersetzt werden sollen. In den Hauptelementen <**DPList**>, <**Connect**> und <**Matrix**> kann ein Knotenname **nn**="[NodeName]" angegeben werden, **nn**=".." stellt sicher, dass die Daten ausschliesslich dem entsprechenden DxNode angefügt werden. Mit **nn**=".." ist es auch möglich, EINE externe Datei für mehrere DxNode zu definieren.

Das Standardverzeichnis für Konfigurationsdateien ist das Arbeitsverzeichnis von DxNode, sofern der Dateiname nicht absolut (d.h. mit Laufwerk und Pfad) angegeben wird (siehe auch ➲ **path**=".."). Der Name kann also mit Laufwerk und Pfad - oder relativ zum Arbeitsverzeichnis mit .. (zwei Punkte) eingegeben werden, dabei zeigt .. auf das übergeordnete Verzeichnis des Arbeitsverzeichnisses. Gültige Einträge sind z.B. C:\Node\Config.xml oder ..\Cnf\Config.xml bzw. Config.xml, wobei hier z.B. C:\Work\Cnf\Config.xml bzw. C:\Work\Config.xml im Arbeitsverzeichnis C:\Work gemeint sind. Die Dateinamen müssen systemkonform sein. Gültige Einträge sind:

Wenn eine extern parametrierte **Datenpunktliste** eingefügt werden soll ...

```
<DxCnfLoc>
   <DPList fn="[FileName]"/>
```

Die lokale <**DPList**> wird beim Hochfahren von DxNode durch den Inhalt in [FileName] vollständig ersetzt. Eine Datei mit einer Datenpunktliste für eine Gruppe würde z.B. so aussehen:

Externe Datei [FileName] wird als Datenpunktliste eingefügt

```
<?xml version="1.0" encoding=".."?>
<!--Datenpunktliste [FileName] -->
<DxCnfExt config_version="..">
   <DPList nn="..">
      <Group gn="..">
         <P n=".." a=".." [..]/><P n=".." a=".." [..]/><P n=".." a=".." [..]/><P n=".." a=".." [..]/>
      </Group>
   </DPList>
</DxCnfExt>
```

Wenn eine extern parametrierte **Verbindung** eingefügt werden soll ...

```
<DxCnfLoc>
   <Connect fn="[FileName]"/>
```

Der entsprechende lokale <**Connect**> wird beim Hochfahren von DxNode durch den Inhalt in [FileName] vollständig ersetzt. Eine Datei mit einer Verbindung für eine Gruppe von Datenpunkten mit Read Only, Bi-direktionalen und Store&Fwd Daten würde z.B. so aussehen:

Externe Datei [FileName] wird als Verbindung eingefügt

```
<?xml version="1.0" encoding=".."?>
<!-- Verbindung [FileName] -->
<DxCnfExt config_version="..">
   <Connect nn=".." cn=".." host=".." port=".." alive=".." [..]>
      <SX gn="..">
         <P n=".." r="=" /><P n=".." r="=" w="=" /><P n=".." r="=" attr="S" />
      </SX>
   </Connect>
</DxCnfExt>
```

Wenn eine extern parametrierte **Kommunikationsmatrix** eingefügt werden soll ...

```
<DxCnfLoc>
   <Matrix fn="[FileName]"/>
```

Die lokale <**Matrix**> wird beim Hochfahren von DxNode durch den Inhalt in [FileName] vollständig ersetzt. Eine Datei mit einer Matrix für eine Verbindung wie oben dargestellt würde z.B. so aussehen:

Externe Datei [FileName] wird als Kommunikationsmatrix eingefügt

```
<?xml version="1.0" encoding=".."?>
<!-- Kommunikationsmatrix [FileName] -->
<DxCnfExt config_version="..">
   <Matrix nn="..">
      <Connect cn="..">
         <SX gn="..">
            <P n=".." r="=" /><P n=".." r="=" w="=" /><P n=".." r="=" attr="S" />
         </SX>
      </Connect>
   </Matrix>
</DxCnfExt>
```

Wenn beliebige extern parametrierte Elemente aus einer Datei eingefügt und aktiviert werden sollen ...

```
<LinkOn>
   <GetFile from="[SourceFileName]" to="[TargetFileName]" />
   <Include fn="[TargetFileName]" />
```

Das Befehlselement <**Include**> ist in <**LinkOn**> sinnvoll, wenn vorerst die gewünschte externe Konfigurationsdatei mit <**GetFile**> heruntergeladen wird (Download). Dies kann zur Laufzeit von DxNode geschehen indem die entsprechende Verbindung zum Rechner, der die Konfigurationsdatei hält, (re)initialisiert wird. Der freistehende Befehl <**Include fn**=".."> wird jedoch immer ausgeführt, auch wenn keine neue Konfiguration heruntergeladen wurde. Um dies zu vermeiden, kann man den Befehl auch direkt als Element in <**GetFile**> einfügen:

```
<LinkOn>
   <GetFile from="[SourceFileName]" to="[TargetFileName]"><Include/></GetFile>
```

In diesem Falle wird die Datei mit [TargetFileName] an den <**Include**> Befehl vererbt und nur dann verarbeitet, wenn eine neue bzw. geänderte Konfiguration heruntergeladen wurde.

Der Include-Befehl modifiziert die Parametrierung zur Laufzeit wobei die Aktuelle Datenpunktliste ggf. mit neuen Datenpunkten erweitert wird (kein Löschen). Verbindungen sowie entsprechende Kommunikationsmatrizen werden vollständig ersetzt, wobei die betroffenen Verbindungen automatisch neu initialisiert werden. Eine Datei mit einer Datenpunktliste und einer Matrix für eine Verbindung wie oben dargestellt würde z.B. so aussehen:

Externe Datei [TargetFileName] wird als Datenpunktliste und Kommunikationsmatrix eingefügt

```
<?xml version="1.0" encoding=".."?>
<!-- Datenpunktliste und Kommunikationsmatrix [FileName] -->
<DxCnfExt config_version="..">
   <DPList nn="..">
      <Group gn="..">
         <P n=".." a=".." [..]/><P n=".." a=".." [..]/><P n=".." a=".." [..]/><P n=".." a=".." [..]/>
      </Group>
   </DPList>
   <Matrix nn="..">
      <Connect cn="..">
         <SX gn="..">
            <P n=".." r="=" /><P n=".." r="=" w="=" /><P n=".." r="=" attr="S" />
         </SX>
      </Connect>
   </Matrix>
</DxCnfExt>
```

from="Source File" to="Target File"

- *Definition und Anwendung*

Eine Quellen- und eine Zieldatei sind im Download-Befehl <**GetFile**> anzugeben. Der Befehl darf nur im Element <**LinkOn**> aufgerufen werden und dient dazu, eine Datei herunterzuladen. Der Download wird automatisch zur Laufzeit von DxNode ausgeführt, nachdem die entsprechende Verbindung aktiviert wurde. Dabei wird eine Quellendatei auf eine Zieldatei kopiert wobei die Dateinamen systemkonform sein müssen. Der Befehl wird z.B. vor einem <**Include**> Befehl ausgeführt um eine externe Konfiguration zur Laufzeit auf den neuesten Stand zu bringen. Gültige Einträge sind:

```
<LinkOn>
   <GetFile from="[SourceFileName]" to="[TargetFileName]"><Include/></GetFile>
```

In diesem Falle wird die Datei mit [TargetFileName] automatisch an den <**Include**> Befehl vererbt und nur dann verarbeitet, wenn eine neue bzw. geänderte Konfiguration heruntergeladen wurde.

```
<LinkOn>
   <GetFile from="[SourceFileName]" to="[TargetFileName]" />
   <Include fn="[TargetFileName]" />
```

In diesem Falle wird der Befehl <**Include fn**=".."> unabhängig von <**GetFile**> immer ausgeführt, auch wenn keine neue Konfiguration heruntergeladen wurde. Dies ist für grosse Dateien ggf. unerwünscht.

gn="Group Name"

- *Definition*

Die Datenpunktliste <**DPList**> kann mit Strukturelement <**Group**> in Gruppen eingeteilt werden. Dabei ist für jedes Strukturelement <**Group**> ein eindeutiger Gruppenname zu definieren mit dem alle Datenpunkte <**P**> einer Gruppe angesprochen werden können. Ein Datenpunkt <**P**> kann nur einer Gruppe zugeordnet werden. Sinnvolle Gruppen bilden z.B. die Datenpunkte einer Einrichtung mit gleichen Eigenschaften wie Read Only, Bi-direktional oder Store&Fwd. Gültige Einträge sind:

```
<DPList>
   <Group gn="[A-Za-z0-9_]">
      <P a=".." n="..">[<E ... />]</P>
```

Der Gruppenname ist vorzugsweise IEC-1131 konform. Die Einschränkungen sind im XML Schema festgelegt. Eine einfache Datenpunktliste <**DPList**> könnte z.B. folgende Gruppen definieren:

```
<DPList>
   <Group gn="Dev1_Rd"><P a=".." n=".." /><P a=".." n=".." /><P a=".." n=".." /></Group>
   <Group gn="Dev1_RW"><P a=".." n=".." /><P a=".." n=".." /><P a=".." n=".." /></Group>
   <Group gn="Dev1_SF"><P a=".." n=".." /><P a=".." n=".." /><P a=".." n=".." /></Group>
<DPList>
```

- *Anwendung*

Der Gruppenname kann zur Auswahl von Datenpunkten <**P**> für eine Subskription <**CX**> oder <**SX**> (Daueraufträge), für einen Single Request <**QX**> (Einzelabfrage) und/oder mit der darunter liegenden Datenpunktauswahl <**P**> verwendet werden. Gültige Einträge sind:

```
<CX [gn="[GroupName|Wildcard]"][attr=".."]> und/oder
<SX [gn="[GroupName|Wildcard]"][attr=".."]> oder
<QX [gn="[GroupName|Wildcard]"][attr=".."]> (nur in Telegramm)
   <P a|n=".." r=".." [w=".."][gn="[GroupName|Wildcard]"][attr=".."][..]/>
```

Zur Selektion mehrerer Gruppen können auch Wildcards (*?) in **gn**=".." verwendet werden. Der Standard Gruppenname ist **gn**="*" (alle Gruppen). Der Gruppenname **gn**=".." im übergeordneten Element wird an die Datenpunktauswahl <**P**> vererbt, wenn dort kein **gn**=".." angegeben ist. Bei einer Überparametrierung d.h. wenn ein Gruppenname in <**P**> angegeben ist, definiert immer dieser die Datenpunktauswahl <**P**>.

Der Ausdruck <**CX gn**="Dev1_Rd"><**P n**="*" **r**="=" /></**CX**> ist also zum Beispiel gleichbedeutend mit <**CX**><**P n**="*" **r**="=" **gn**="Dev1_Rd"/></**CX**> und selektiert die Datenpunkte der Gruppe Dev1_Rd. Der Ausdruck <**CX gn**="Dev1_Rd"><**P n**="*" **r**="=" **gn**="*"/></**CX**> würde hingegen die Datenpunkte aller Gruppen auswählen und wäre einfacher mit <**CX**><**P n**="*" **r**="="/></**CX**> darzustellen.

Der Gruppenname kann auch zur Auswahl von Datenpunkten <**P**> in einem Steuerelement <**Link...**> verwendet werden. Gültige Einträge sind:

```
<Link1st> und/oder
<LinkOn> und/oder
<LinkOff>
   <P a|n=".." [gn="[GroupName|Wildcard]"]><D|E|e [v=".."][..]/></P>
```

Zur Selektion mehrerer Gruppen können auch Wildcards (*?) in **gn**=".." verwendet werden. Der Standard Gruppenname ist **gn**="*" (alle Gruppen).

host="Host Name or IP-Address"

- *Definition*

Für aktive Anschlüsse <**Connect**> ist der Rechnername oder die IP-Adresse des anzusprechenden Partnerknotens jeweils zusammen mit dessen <**Daemon**> Port-Adresse bzw. Service-Nr anzugeben, d.h. **host**=".." und **port**=".." definieren als Paar den Partner mit Rechner (Host/IP) und Zugangsport zum DxNode (Daemon Port). Gültige Einträge sind:

<**DxCnfLoc**> oder <**DxCnfExt**>
<**Connect cn**=".." **host**="[Host/IP[:Port]][;Host/IP[:Port]][;...]" **port**="[DaemonPort]" [..]>

Dabei können mit **host**=".." eine oder mehrere redundante Rechner oder IP-Adressen, getrennt durch Strichpunkt (Semicolon), angegeben werden z.B. **host**="Host1;Host2;Host3". In diesem Falle gilt für alle Hosts den gleichen **port**="..".

Im Weiteren kann jedem Host ein eigener Port, getrennt durch Doppelpunkt (Colon), zugeordnet werden z.B. **host**="Host:DaemonA;Host:DeamonB". Dies ist notwendig, wenn verschiedenen Ports z.B. auf dem gleichen Rechner angesprochen werden sollen. In diesem Fall soll der Eintrag **port**=".." weggelassen werden.

Mit der Verbindungsherstellung oder bei Verbindungsverlust wird jeweils die Liste der Hosts von links nach rechts und wieder von vorne abgearbeitet, bis eine lauffähige Verbindung erstellt und mit <**LinkOn**> angezeigt werden kann. Die Verbindung bleibt dann solange erhalten, bis sie z.B. mittels internem Datenpunkt [Prefix][ConnectName].cmdio.state abgebrochen wird, um eine neue Verbindung zu suchen.

Ein bestimmter Host kann auch mittels internem Datenpunkt [Prefix][ConnectName].cmdio.active_host direkt angewählt werden. Dabei ist der Wert z.B. auf 1, 2 oder 3 zu stellen und den entsprechenden **host**="Host1;Host2;Host3" auszuwählen. Wird ein Host nicht gefunden, so wird der nächste Host in der Liste gesucht. Nach durcharbeiten der Liste wird wieder vorne von angefangen. Eingabewerte:

0 = Re-Initialisierung, beginne Hostsuche von vorne
1..X = Setze Hostsuche auf Position 1..X der Liste

Ein aktiver Anschluss <**Connect**> kann z.B. zwei redundante DxNode anbinden, die nur EINE redundante Anlage darstellen. Das redundante System soll die Verbindung im passiven Rechner mittels [Prefix][ConnectName].cmdio.state sperren und nur dann freigeben, wenn dieser Rechner mit aktuellen Daten verfügbar ist. (Wenn ein passiver Rechner keine brauchbaren Daten liefert, dann MUSS diese Verbindung gesperrt werden, weil ja die Verbindung ggf. mit dem passiven Rechner bestehen bleibt). Wenn zwei Rechner die Daten parallel zur Verfügung halten, dann müssen die Verbindungen zum gemeinsamen DxNode nicht gesperrt werden (Parallelbetrieb).

Store&Fwd Meldungen werden nur in demjenigen DxNode erzeugt, wo die Server Subskription <**SX**> mit **attr**="S" parametriert ist. Das ist für redundante Verbindungen normalerweise der passive Server (ohne **host**=".." **port**="..") und bedeutet, dass vom aktiven Client (mit **host**=".." **port**="..") keine Store&Fwd Meldungen über die gleiche Verbindung geschickt werden können. Wenn Store&Fwd Meldungen in beide Richtungen gefordert sind, dann sind zwei Verbindungen zu definieren.

- *Anwendung*

Mit dem Standardwert für **port**="7581/tcp" wird z.B. ein Verbindungsaufruf mit Name "ConnSys1" zu einer gleichnamigen passiven Verbindung im Rechner "192.37.36.1" so parametriert:

<**Connect cn**="ConnSys1" **host**="192.37.36.1" **port**="7581/tcp"><**Switch**/></**Connect**>

Wenn zusätzlich ein redundanter Rechner "192.37.36.2" parametriert werden soll, dann wird obiger Ausdruck für zwei Rechner zu ...

<**Connect cn**="ConnSys1" **host**="192.37.36.1;192.37.36.2"
port="7581/tcp"><**Switch**/></**Connect**>

Eine Verbindung <**Connect**> kann auch zum eigenen DxNode hergestellt werden. In diesem Falle ist **host**="localhost" oder **host**="127.0.0.1" einzusetzen. Das kann z.B. für Testzwecke nützlich sein oder um Datenpunkte mit neuen Namen und Eigenschaften zu duplizieren um sie anderen Partnern oder Verbindungen zur Verfügung zu stellen.

Eine solche Verbindung kann direkt zum eigenen <**Daemon**> hergestellt werden d.h. es wird nicht mit <**Switch**> auf eine parametrierte passive Verbindung geschaltet:

<**Connect cn**="LocalDaemon" **host**="localhost" **port**="7581"></**Connect**>

Wenn auf eine parametrierte Verbindung im eigenen DxNode geschaltet werden soll, dann muss im Element <**Switch**> der Name der passiven Verbindung stehen. Weil eine Verbindung nicht zu sich selbst hergestellt werden kann, sind zwei verschiedene Verbindungen bzw. Namen anzusprechen:

<**Connect cn**="LocActive" **host**="localhost" **port**="7581"><**Switch cn**="LocPassive" /></**Connect**>

Der Partner Host-Name oder die IP-Adresse kann im aktiven DxNode zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].cmdio.**host** abgefragt oder verändert werden. Die parametrierte Vorgabe **host**=".." wird dabei nicht verändert. Die IP-Adresse des Partners kann zusätzlich über den ➲ internen Datenpunkt [Prefix][ConnectName].cmdio.host_ip abgefragt werden.

i="Index" siehe auch v=".."

- *Definition*

Der Meldungsindex markiert allgemein eine bestimmte Nachricht bzw. einen bestimmten Wert **v**=".." und kann bei Bedarf in der <**DPList**> oder in den Steuerelementen <**Link**...> pro Datenpunkt-Auswahl <**P**> mit <**D**|**E**|**e i**=".." /> vorgegeben werden. Gültige Einträge sind:

<**D**|**E**|**e** [**v**=".."] **i**="[-2147483648...+2147483647]" [..]/>

Der Standardwert ist **i**="0". Der Index ist anwendungsseitig zu definieren und anzuwenden. Ein Index kann z.B. von der Server-Anbindung als <**E [..] i**="123456" /> verschickt - und von einer bestimmten Client-Anwendung mit <**e [..] i**="123" /> bestätigt werden, wobei "123456" als Unique Number die eindeutige Identifikation ermöglicht.

- *Anwendung*

Konfigurationsbeispiel: Für eine Anbindung sei eine Verbindung <**Connect**> mit Adressen **a**=[Addr] in DxNode wie folgt parametriert:

```
<Connect cn="..">
   <CX><P a="[Addr]" r="=" /></CX>            (Client Subskription für Daten von Anbindung lesen)
   <SX><P a="CheckAddr" r="[Addr]" attr="+a-v-t-q-f-s+i-u-x" /></SX>      (Server Subskription)
</Connect>
```

Die <**CX**> ist die Standardkonfiguration für "Daten von Anbindung=Server lesen". Mit der <**SX**> werden nun alle [Addr] von DxNode auf eine fixe "CheckAddr" in die Anbindung=Client zurückgemeldet. Mit der Definition **attr**=".." werden nur die Ursprungsadresse [Addr] und der [Index] an die Anbindung gespiegelt, d.h. anderen Daten werden nicht übermittelt. Ein Telegramm von der Anbindung=Server ...

<**X0**><**P a**="[Addr]"><**E v**=".." **i**="[Index]" [..]/></**P**></**X0**>

... wird von DxNode als Rückmeldung an Anbindung=Client gespiegelt:

<**X0**><**P a**="CheckAddr"><**E a**="[Addr]" **i**="[Index]" /></**P**></**X0**>

← Ausgangstelegramm
Anbindung
→ Rückmeldetelegramm

Die Anbindung erhält also für jedes versendete Ereignis ein Rückmeldetelegramm auf eine fixe Adresse "CheckAddr" wobei der Inhalt die Ursprungsadresse [Addr] mit dem von der Anbindung versendeten [Index] als Quittung darstellt.

ip="IP Address" (DxMonitor)

- *Definition und Anwendung*

DxMonitor erlaubt das Überwachen und Aufzeichnen von Verbindungen in DxNode. Zur Identifikation namensloser Verbindungen übermittelt DxNode die IP Adresse an DxMonitor. DxMonitor zeichnet dann die Telegramme in der Log Datei wie folgt auf:

<**S0 t**=".." **ip**="[Partner]" [..]><**X0**> **t**=".." [..]</**X0**></**S0**> (DxNode sendet <**X0**> an Partner)
<**R0 t**=".." **ip**="[Partner]" [..]><**X0**> **t**=".." [..]</**X0**></**R0**> (DxNode empfängt <**X0**> von Partner)

Zulässiger Wert für [Partner] ist eine gültigen IP Adresse z.B. **ip** ="[50.30.17.44]".

☞ Attribut **ip** =".." kommt nur in Meldungen an DxMonitor vor.
Es kann nicht konfiguriert werden und kommt daher in den Konfigurationsdateien nicht vor.

ip_list="IP Address List" (DxMonitor)

- *Definition und Anwendung*

DxMonitor erlaubt das Überwachen und Aufzeichnen mehrerer Verbindungen in DxNode. Hierzu kann eine Liste von IP Adressen als Befehl an DxNode geschickt werden:

<**X0**><**Connect ip_list**="[xx;yy;zz]" [**verbose**=".."] [..]/>[<**CX**>|<**SX**>]</**X0**> (Befehl an DxNode)
<**Y0**><**Connect ip_list**="[xx;yy;zz]" [**verbose**=".."] [..]/>[<**CX**>|<**SX**>]</**Y0**> (DxLog Datei)

In der Log Datei wird der Befehl als <**Y0**> Telegramm ausgegeben. Zulässiger Wert [xx;yy;zz] ist eine Liste von gültigen IP Adressen, getrennt durch Semicolon (;) z.B. **ip_list**="[50.31.0.200;50.30.17.44]".

☞ Attribut **ip_list**=".." kommt nur in Meldungen an DxMonitor vor.
Es kann nicht konfiguriert werden und kommt daher in den Konfigurationsdateien nicht vor.

l="Length of Message" (reserved)

- *Definition und Anwendung*

Die allgemeine Form der XML-Telegramme auf TCP/IP Ebene (ASCII, Stream-Socket) ist ...

hhhhhhhh<**Telegram [..]**><[..]></**Telegram**>

Dabei stellt der Header "hhhhhhhh" die Länge der XML Meldung in [bytes] als Hexzahl (fix 8-stellig) dar, d.h. von incl. StartTag <**Telegram**> bis incl. EndTag </**Telegram**>. Für Telegramme <**X0**> kann diese Länge als Attribut, umgerechnet auf eine Dezimalzahl (fix 10-stellig), übermittelt werden. Gültige Darstellungen sind:

hhhhhhhh<**X0** [**l**="[0000000027..2147483647"][**t**=".."][..]><[..]></**X0**>

Hier ist die Länge der Meldung **l**=".." in [bytes] zusätzlich als Dezimalzahl (fix 10-stellig) dargestellt, d.h. von und incl. <**X0**> bis incl. </**X0**>. Das theoretisch kürzeste XML Telegramm <**X0**> mit nur einem Element <**P**> ohne Daten hätte z.B. genau 27 Zeichen und würde auf der TCP/IP Ebene als ASCII-Zeichenfolge mit 0000001B<**X0 l**="0000000027"<**P**/></**X0**> übermittelt.

Um das Parsen zu vereinfachen wird das Attribut - wenn definiert - an erster Stelle eingefügt, d.h. die Länge kann z.B. nach dem Einlesen der ersten 17 Zeichen (ohne Header) direkt als Dezimalzahl aus **l**=".." gelesen werden.

☞ Attribut **l**=".." wird z.Z. nicht unterstützt.

max_conn="Max Number of Connects"

- *Definition und Anwendung*

Die maximale Anzahl der gleichzeitig aktiven Verbindungen kann pro <**Daemon**> Element definiert werden. Gültige Einträge sind:

```
<DxCnfLoc>
   <Daemon dn=".." port=".." [max_conn="[1..255]"][..]/>
```

Der Standardwert ist **max_conn**="10", dies bedeutet, dass gleichzeitig bis zu 10 externe Teilnehmer über den Zugangsport <**Daemon**> kommunizieren können. Beidseitig parametrierte Verbindungen belasten den Zugangsport jedoch ggf. nur bis die Verbindung hergestellt - d.h. an den passiv parametrierten Teilnehmer vermittelt ist.

Die maximale Anzahl der gleichzeitig aktiven Verbindungen kann zur Laufzeit über den ➲ internen Datenpunkt [Prefix][DaemonName].value.**max_conn** abgefragt werden. Ein Bit-Muster der aktiven bzw. inaktiven Verbindungen ist über [Prefix][DaemonName].value.connects ablesbar. Die Darstellung ist ein String z.B. "1111000000" bei 4 aktiven Verbindungen und **max_conn**="10".

max_telegr_length="Max Length of Telegram" (reserved)

- *Definition und Anwendung*

Die maximale Länge in [KB] der Telegramme <**X0**> mit Elementdaten <**P**><**D**|**E**|**e [..]**/></**P**> kann vorgegeben werden. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." [max_telegr_length="[1..2097152]"][..]/>
```

Der Standardwert ist **max_telegr_length**="2097152", dies bedeutet, dass praktisch ein unbegrenzter Datenstrom (bis 2 Gigabyte) als EIN Telegramm <**X0**> übermittelt werden könnte.

☞ Die Länge ist z.Z. fix auf 128 KB eingestellt und kann nicht durch die Konfiguration eingestellt werden.

Die Telegrammlänge zusammen mit dem Lese- und dem Schreibintervall, definieren eine maximale Übertragungsrate pro Verbindung, ➲ siehe auch **rd_interval**=".." **wr_interval**="..".

msg="Clear Text Message"

- *Definition*

Fehler-, Warnungs- und Informationsmeldungen von DxNode werden als spezielle Error-Telegramme <**E1**>, <**E2**> und <**E4**> mit einer Klartextmeldung in XML-Notation ausgegeben oder aufgezeichnet. Der Inhalt von Attribut **msg**=".." ist jeweils die Klartextmeldung. Gültige Darstellungen sind:

Fehlermeldungen (Error Level 1) ...

```
<E1 t=".." [..] msg="[anyText]" />
```

Fehlermeldungen <**E1**> (Failure) bedeuten, dass die angeforderte Funktion von DxNode nicht erfüllt werden konnte, z.B. Konfigurationsdatei nicht gefunden oder bei Zugriff auf korrupte interne Daten.

Warnungsmeldungen (Error Level 2) ...

```
<E2 t=".." [..] msg="[anyText]" />
```

Warnungsmeldungen <**E2**> (Warning) zeigen wichtige Ereignisse wie Trace Messages aus den Steuerelementen <**Link1st**>, <**LinkOn**>, <**LinkOff**> z.B. <**E2 t**=".." **msg**="*** Link Initialization ***"/> oder sie deuten auf mögliche Fehler hin die jedoch keine unmittelbaren Folgen für DxNode haben z.B. ungültiges Telegramm eines Teilnehmers.

Informationsmeldungen (Error Level 4) ...

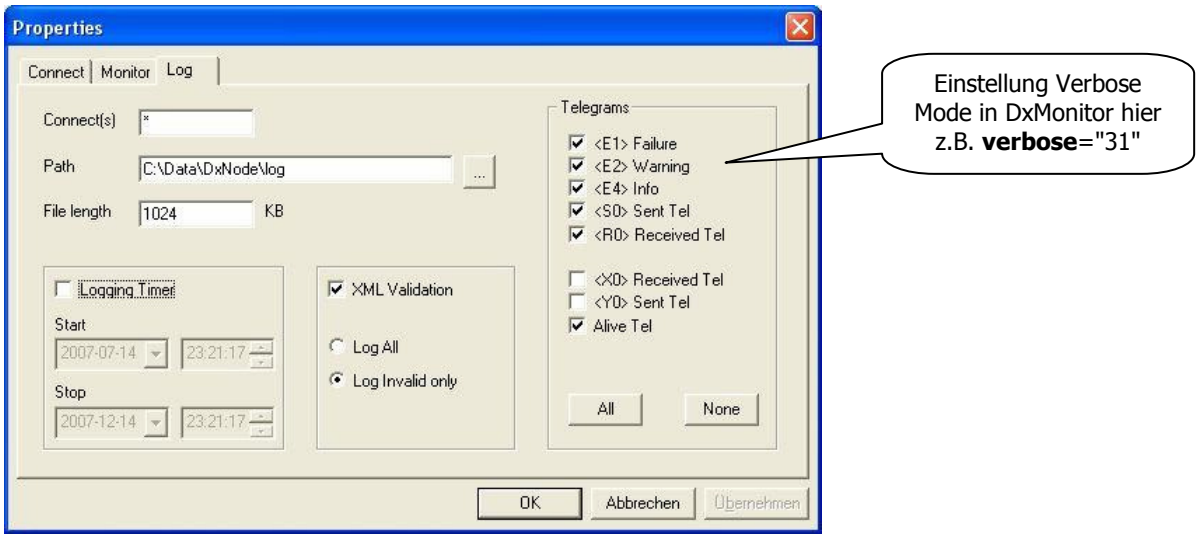
```
<E4 t=".." [..] msg="[anyText]" />
```

Informationsmeldungen <**E4**> (Information) zeigen zulässige Operationen an die keine direkten Folgen für DxNode haben z.B. neuer Datenpunkt <**P**> anlegen.

- *Anwendung*

Fehlermeldungen <**E1**> und Warnungsmeldungen <**E2**> werden in der Log-Datei [NodeName].log bzw. [NodeName].bak im Arbeitsverzeichnis (siehe auch ➲ **path**="..") von DxNode aufgezeichnet. Der Dateiname ist definiert durch den Eintrag **nn**="[NodeName]" in Element <**Node**>.

Alle Fehler-, Warnungs- und Informationsmeldungen <**E1**>, <**E2**> und <**E4**> können mit DxMonitor unter Berücksichtigung von bestimmten Trefferkriterien ausgewählt, beobachtet und aufgezeichnet werden. Mit dem Wert in Attribut **verbose**=".." kann bei der Verbindungsherstellung die Meldungstiefe festgelegt werden: "0" für keine, "1" für <**E1**>, "2" für <**E2**> und "4" für <**E4**> Meldungen, zudem können die Werte auch kombiniert werden z.B. "7" für alle Meldungen <**E1**>, <**E2**> und <**E4**>.



Der DxMonitor erlaubt die Darstellung zusätzlicher Daten-Transaktionen wie folgt ...

<**S0 t**=".." [..]><**X0**> **t**=".." [..]</**X0**></**S0**>	(DxNode sendet Telegramm <**X0**> an Partner)
<**R0 t**=".." [..]><**X0**> **t**=".." [..]</**X0**></**R0**>	(DxNode empfängt Telegramm <**X0**> von Partner)
<**Y0 t**=".." [..]></**Y0**>	(DxMonitor schreibt Telegramm <**X0**> zum DxNode)
<**X0 t**=".." [..]></**X0**>	(DxMonitor liest Telegramm <**X0**> von DxNode)

n="Name (network)" siehe a=".."

- *Definition und Anwendung*

Der Netzwerk Name und/oder die Lokale Adresse der einzelnen Datenpunkte <**P**> werden in der <**DPList**> definiert und sind darstellbar als zwei gleichwertige Adressräume (Lokal und Netzwerk) im lokalen DxNode. Für gültige Eingaben ➲ siehe **a**="Address (application)".

nid="Node ID" sid="Sender ID" rid="Receiver ID"

- *Definition und Anwendung*

In Telegrammen <**X0**> kann eine Sender- und Empfänger-Identifikation mit **sid**=".." und **rid**=".." übermittelt werden. Der Inhalt wird im Element <**Node**> des Sender- und Empfänger-Knotens mit dem Eintrag für die Knoten-Identifikation bestimmt. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." [nid="[anyExceptSpecialChar]"][..]/>
```

Die Knoten-Identifikation soll nur sichtbare Zeichen, jedoch keine Wildcards (*?), XML-Steuerzeichen (<>'&") oder reservierte Zeichen (;#~^|) enthalten. Die Einschränkungen sind im XML Schema festgelegt.

Die Knoten-Identifikation wird in Telegrammen <**X0**> zwischen zwei DxNode automatisch als Sender- bzw. Empfänger-Identifikation mit **sid**=".." und **rid**=".." mitgeführt, sofern im entsprechenden DxNode eine Identifikation **nid**=".." definiert ist:

```
<X0> t=".." [sid="[SenderNodeID]"][rid="[ReceiverNodeID]"][..]>
```

Fehlt die Angabe **nid**=".." in DxNode, dann wird das entsprechende Attribut nicht übermittelt. Die Sender- und Empfänger-Identifikation haben nur informativen Charakter und werden zur Übertragung nicht benötigt, weil die Verbindungen mit ihrem Namen **cn**=".." eindeutig identifizierbar sind. System-Anbindungen können daher auch eine fest programmierte oder keine Identifikation liefern. Andererseits ist eine Identifikation für eine Verbindung zu DxMonitor sinnvoll, weshalb Error- und Beobachtungstelegramme <**E1**>, <**E2**>, <**E4**>, <**S0**> und <**R0**> Knotennamen **nn**=".." und/oder den Verbindungsnamen **cn**=".." zur Identifikation mitführen.

nn="Node Name"

- *Definition*

Ein netztwerkweit eindeutiger Knotenname soll für jeden DxNode im Element <**Node**> definiert werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**="[A-Za-z0-9_]" [**prefix**=".."][..]/>

Der Knotenname ist vorzugsweise IEC-1131 konform da er zur Bezeichnung von internen Datenpunkten verwendet wird, die Einschränkungen sind im XML Schema festgelegt.

- *Anwendung*

Für die Elemente <**DPList**>, <**Connect**>, <**Matrix**> und <**Include**> kann eine separate bzw. externe Konfigurationsdatei definiert werden, um dort parametrierbare Elemente zu definieren bzw. zu ersetzen. Die externe Datei wird in den o.g. Elementen referenziert, ➲ siehe auch **fn**="[FileName]".

Der Inhalt der externen Datei muss identisch mit einer vergleichbaren lokalen Konfiguration sein. Die Datei kann eine Konfigurationsversion **config_version**=".." enthalten und soll in den Hauptelementen <**DPList**>, <**Connect**> und <**Matrix**> Knotennamen **nn**="[NodeName]" sowie alle Elemente aufführen, die ersetzt werden sollen. Mit **config_version**=".." können die Daten ggf. identifiziert werden und der Knotenname **nn**=".." stellt sicher, dass die eingefügten Daten zum entsprechenden DxNode gehören, d.h. es kann auch nur EINE externe Datei für mehrere Knoten definiert werden. Der Dateiname muss systemkonform sein. Gültige Darstellungen für die externe Konfigurationsdatei sind:

Referenzierte Datei [FileName] als externe Konfigurationsdatei
<?**xml version**="1.0" **encoding**=".."?> <!-- Externe Konfigurationsdatei [FileName] --> <**DxCnfExt config_version**="...">   <**DPList nn**="[NodeName]" [..]> ... </**DPList**> und/oder   <**Connect cn**="..." **nn**="[NodeName]" [..]> ... </**Connect**> und/oder   <**Matrix nn**="[NodeName]" [..]> ... </**Matrix**> </**DxCnfExt**>

Der Knotenname [NodeName] wird zur Bezeichnung ➲ interner Datenpunkte verwendet, dabei kann mit <**Node prefix**=".." [..]/> eine linksbündige Erweiterung eingefügt werden:

Lokale Adresse a=".."	**Beschreibung**
[Prefix][NodeName].cmdio.sss	Befehls-Ein/Ausgabe (Bi-direktional Read/Write)
[Prefix][NodeName].value.sss	Wert-Ausgabe/Anzeige (Read Only)
[Prefix][NodeName].perf.sss	Performance-Anzeige (Read Only)

... wobei sss = Sub-Element-Bezeichnung = Attribut-Name bei parametrierbaren Items
(Performance-Werte werden periodisch gemäss **perf_cycle**=".." updated/ausgegeben)

Fehlermeldungen <**E1**> und Warnungsmeldungen <**E2**> werden automatisch in der Log-Datei [NodeName].log bzw. [NodeName].bak im Verzeichnis von DxNode aufgezeichnet, ➲ siehe **msg**="..".

Der Knotenname [NodeName] bezeichnet auch die ggf. für DxNode anzulegende interne Datei für zu speichernde Daten zur Wiederherstellung von gewissen Eigenschaften z.B. Zeitstempel des zuletzt empfangenen Telegrammes für Store&Fwd Meldungen.

path="Working Directory"

- *Definition und Anwendung*

Mit einer Pfadangabe wird das Arbeitsverzeichnis ersetzt oder umgeschaltet. Ohne Pfadangabe arbeitet DxNode im Verzeichnis aus dem er gestartet wurde. Das Arbeitsverzeichnis enthält die DxNode Log-Datei [NodeName].log sowie alle Store&Fwd Sicherungsdateien. Es ist auch das Standardverzeichnis für Konfigurationsdateien, sofern diese nicht mit dem absoluten Dateinamen (d.h. mit Laufwerk und Pfad) angegeben werden. Wenn das entsprechende Verzeichnis im System nicht vorhanden ist, wird es von DxNode hergestellt. Die Pfadangabe muss systemkonform sein. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." path="[Path]" [..]/>
```

Der Standardwert ist das Start-Arbeitsverzeichnis von DxNode. Die Angabe kann absolut (d.h. mit Laufwerk und Pfad) - oder relativ zum Arbeitsverzeichnis mit (..) eingegeben werden. Dabei zeigt (..) auf das übergeordnete Verzeichnis des Arbeitsverzeichnisses.

Das nebenstehende Beispiel zeigt die Einträge für eine Verknüpfung zum Start von DxNode vom Desktop mit folgenden Start-Verzeichnissen:

- Programmverzeichnis C:\Node\Bin
- Konfigurationsverzeichnis C:\Conf
- Arbeitsverzeichnis C:\Work

Mit der Eingabe von **path**="..\Log" würde das Verzeichnis "Log" auf der gleichen Ebene wie das Arbeitsverzeichnis (angelegt und) verwendet d.h. das Arbeitsverzeichnis wäre zur Laufzeit C:\Work\Log.

Mit der Eingabe von **path**="C:\Temp" würde das Arbeitsverzeichnis C:\Temp (angelegt und) verwendet d.h. das Arbeitsverzeichnis C:\Work gilt nur beim Starten und würde für die Laufzeit ersetzt.

perf_cycle="Performance Check Interval"

- *Definition*

Die Aktualisierungsrate der Statistik in [ms] kann im Element <**Node**> pro DxNode festgelegt werden. Sie definiert das Zeitintervall zur Nachführung der internen Datenpunkte mit Performance-Zahlen. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." perf_cycle="[0|500..65535]" [..]/>
```

Der Standardwert ist **perf_cycle**="0" [ms], d.h. die Nachführung ist ausgeschaltet. Eine Eingabe von **perf_cycle**="7500" bedeutet, die Performance-Zahlen werden alle 7.5 [sec] nachgeführt.

- *Anwendung*

Die Performance-Zahlen können über folgende ➲ internen Datenpunkte beobachtet werden:

Lokale Adresse a=".."	**Beschreibung**
[Prefix][ConnectName].perf.sss	Performance-Anzeige (Read Only)
[Prefix][NodeName].perf.sss	Performance-Anzeige (Read Only)

... wobei sss = Sub-Element-Bezeichnung = Attribut-Name bei parametrierbaren Items
(Performance-Werte werden periodisch gemäss **perf_cycle**=".." updated/ausgegeben)

Die Aktualisierungsrate der Statistik kann zur Laufzeit über den ➲ internen Datenpunkt [Prefix][NodeName].cmdio.**perf_cycle** verändert werden. Die parametrierte Vorgabe **perf_cycle**=".." wird dabei nicht verändert.

port="Port Addr or Service-Nr"

- *Definition*

Innerhalb desselben Rechners ist für jeden <**Daemon**> eine eigene Port-Adresse bzw. Service-Nr zu definieren. Über dieses so genannte "Listener Port" (Horcher) empfängt DxNode Anfragen von anderen Teilnehmern und führt entsprechende "Services" (Dienste) aus z.B. Subskription <**CX**> oder <**SX**> (Dauerauftrag für Datenübermittlung). Der Listener Port ist der einzige Zugangspunkt an DxNode, über den sich alle Teilnehmer anmelden müssen, wenn nur ein <**Daemon**> für DxNode definiert ist (es sind jedoch mehrere Daemone pro DxNode und mehrere DxNode pro Rechner definierbar). Mit dem Zugriffsschutz des Listener Ports, kann DxNode auf einfache Weise vor unerwünschtem Zugang geschützt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Daemon dn**=".." **port**="[A-Za-z0-9_/]" [..]>

Der Standardwert ist **port**="7581/tcp", d.h. dieser Port wird normalerweise von allen Teilnehmern angerufen, um mit dem lokalen DxNode zu kommunizieren. Wenn mehrere <**Daemon**> oder DxNode im selben Rechner parametriert bzw. installiert sind, dann müssen die entsprechenden Daemon Ports unterschiedlich bezeichnet werden.

- *Anwendung*

Für aktive Anschlüsse <**Connect**> ist die Listener Port-Adresse bzw. Service-Nr des passiven Partners immer zusammen mit dem Rechner anzugeben, d.h. **host**=".." und **port**=".." definieren den anzusprechenden Partner mit Rechnername (Host) und Zugangsport (Port). Gültige Einträge sind:

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **host**="[Host/IP]" **port**="[Port]" [..]>

Mit dem Standardwert **port**="7581/tcp" könnte z.B. ein Verbindungsaufruf zu Rechner "System1" so parametriert werden: <**Connect cn**="ConnSys1" **host**="System1" **port**="7581/tcp" [..]>.

Der Port kann, getrennt durch einen Doppelpunkt (Colon), auch direkt dem Hostnamen angefügt werden (siehe auch ➲ **host**=".."). Dies ist notwendig, wenn verschiedenen Ports z.B. auf dem gleichen Rechner angesprochen werden sollen. Gültige Einträge sind:

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **host**="[Host/IP[:Port]][;Host/IP[:Port]][;...]" [..]>

In diesem Fall soll der Eintrag **port**=".." weggelassen werden.

Die Listener Port-Adresse bzw. Service-Nr kann für jeden Daemon zur Laufzeit über den ➲ internen Datenpunkt [Prefix][DaemonName].value.**port** abgefragt werden.

Die Partner Port-Adresse bzw. Service-Nr kann für jede Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].cmdio.**port** abgefragt oder verändert werden. Die parametrierte Vorgabe **port**=".." wird dabei nicht verändert.

prefix="Internal DP Name Prefix"

- *Definition und Anwendung*

Präfix zur Erweiterung der internen Datenpunkt-Bezeichnung. Gültige Einträge sind:

```
<DxCnfLoc>
   <Node nn=".." prefix="[A-Za-z0-9_]" [..]/>
```

Standardwert ist kein Präfix (Null-String). Das Präfix ist vorzugsweise IEC-1131 konform da es zur Bezeichnung von internen Datenpunkten verwendet wird, die Einschränkungen sind im XML Schema festgelegt.

Mit Hilfe des Präfix ist es möglich, bei zwei oder mehreren DxNode die gleichen Namen für **nn**="..", **dn**=".." bzw. **cn**=".." zu verwenden und trotzdem pro DxNode individuelle Datenpunkt-Namen für die internen Datenpunkte zu erhalten. Dies ist insbesondere für beidseitig parametrierte Verbindungen wichtig, weil die Verbindungsnamen **cn**=".." vorzugsweise in beiden Partnerknoten identisch sein sollen. D.h. eine bestimmte Verbindung kann in beiden Partnersystemen gleich benannt werden, mit dem Präfix bleiben die Datenpunkt-Namen in den beiden Partner unterschiedlich.

Falls definiert, werden alle internen Datenpunkt-Namen linksbündig um diese Bezeichnung erweitert, z.B. würde **prefix**="Node1_" folgende ➲ internen Datenpunkte definieren:

Namen Definition	Interne DP Namen ohne Präfix	Interne DP Namen mit Präfix
nn="NodeName"	NodeName.value....	Node1_NodeName.value....
dn="DaemonName"	DaemonName.value...	Node1_DaemonName.value...
cn="ConnectName"	ConnectName.value...	Node1_ConnectName.value...

pwd="Password"

- *Definition*

DxNode kann so konfiguriert werden, dass nur mittels Passwort auf den Zugangsport des <**Daemon**> zugegriffen werden kann. Zu diesem Zweck wird für den <**Daemon**> ein [Passwort] parametriert:

```
<DxCnfLoc>
   <Daemon dn=".." port=".." pwd="[Password]" [..]>
```

Zulässige Werte für **pwd**=".." sind alle für XML gültigen Zeichen.

- *Anwendung*

Für passive und aktive Anschlüsse <**Connect**> kann ein [Passwort] parametriert werden, das mit demjenigen des <Daemon> oder dem Partner <Connect> übereinstimmen muss:

```
<DxCnfLoc>
   <Connect cn=".." [host=".."] [port=".."] pwd="[Password]" [..]>
```

Zulässige Werte für **pwd**=".." sind alle für XML gültigen Zeichen.

☞ Passworte werden aus Sicherheitsgründen NICHT ÜBERMITTELT.
Der Datenverkehr für Passwort geschützte Verbindungen wird automatisch encryptet, ist also unlesbar.
Bei inkorrektem oder fehlendem Passwort wird die Verbindung abgebrochen.

q="Quality" siehe auch v=".."

- *Definition und Anwendung*

Die Qualität gilt für Mess- und Sollwerte **v**="..", sie kann in der <**DPList**> oder in Elementen <**Link**..> pro Datenpunkt-Auswahl <**P**> mit <**D**|**E**|**e q**=".." /> vorgegeben werden. Gültige Einträge sind:

<**D**|**E**|**e** [**v**=".."] **q**="[gT|gU|gLO|g|uSN|uEX|uSA|uLV|u|bWD|bOS|bCF|bLV|bSF|bDF|bNC|bCE|b]" [..]/>

Der Initialisierungswert ist **q**="bWD" (bad Waiting for Initial Data) und wird beim Starten von DxNode automatisch gesetzt. Mit Telegrammen <**X0**><**P**> übermittelte Messwerte ohne Qualität, werden bei Empfang in DxNode automatisch auf **q**="g" (good) gesetzt. Die Qualität entspricht einer erweiterten OPC Quality und ist ein enumerated Wert (string) aus folgender Tabelle:

Quality	**Enum**	**Bedeutung gemäss OPC**	**Beschreibung für Wert v**=".."
q="b"	0	bad, Non-specific	schlecht, unbrauchbar ohne weitere Detailangabe
q="bCE"	4	bad Configuration Error	schlecht, Konfigurationsfehler
q="bNC"	8	bad Not Connected	schlecht, der Eingang ist nicht verbunden
q="bDF"	12	bad Device Failure	schlecht, ein Gerätefehler wurde detektiert
q="bSF"	16	bad Sensor Failure	schlecht, der Geber ist ausgefallen
q="bSFL"	17	bad Sensor Failure, Low Limit	schlecht, Geber ausgefallen mit unterem Grenzwert
q="bSFH"	18	bad Sensor Failure, High Limit	schlecht, Geber ausgefallen mit oberem Grenzwert
q="bSFC"	19	bad Sensor Failure, Constant	schlecht, Geber ausgefallen mit Konstante
q="bLV"	20	bad Last Known Value	schlecht, letzter bekannter Wert nach Übertragungsfehler
q="bCF"	24	bad Communication Failure	schlecht, Kommunikationsfehler, letzter Wert fehlt
q="bOS"	28	bad Out Of Service	schlecht, nicht verfügbar, Messung blockiert oder inaktiv
q="bWD"	32	bad Waiting for Initial Data	schlecht, warten auf erste Daten, Default beim Starten
q="u"	64	uncertain, Non-specific	unzuverlässig, ungewiss, unglaubhaft ohne Detailangabe
q="uLV"	68	uncertain Last Usable Value	unzuverlässig, letzter brauchbarer "alter" Wert
q="uSA"	80	uncertain Sensor Not Accurate	unzuverlässig, Geber ungenau, ausserhalb Kalibrierung
q="uSAL"	81	uncertain Sensor Low Limit	unzuverlässig, Geber blockiert mit unterem Grenzwert
q="uSAH"	82	uncertain Sensor High Limit	unzuverlässig, Geber blockiert mit oberem Grenzwert
q="uEX"	84	uncertain Eng. Unit Exceeded	unzuverlässig, Messwert-Bereichsüberschreitung
q="uEXL"	85	uncertain EU Exc. Low Limit	unzuverlässig, Bereichsüberschreitung unterer Grenzwert
q="uEXH"	86	uncertain EU Exc. High Limit	unzuverlässig, Bereichsüberschreitung oberer Grenzwert
q="uEXC"	87	uncertain EU Exc. Constant	unzuverlässig, Bereichsüberschreitung mit Konstante
q="uSN"	88	uncertain Sub-Normal	unzuverlässig, bei mehreren Quellen fehlen welche
q="g"	192	good, Non-specific	gut, Standardeinstellung wenn Qualitätsangabe fehlt
q="gLO"	216	good Local Override	gut, wurde lokal überschrieben bzw. auf Ersatzwert gestellt
q="gT"	**240	good In Transition	gut, Wert verändert od. Befehl, noch nicht zurückgemeldet
q="gU"	**244	good Unacknowledged	gut, Unquittierte Meldung, Quittierung mit **q**="gT"

Interne Enum Werte gemäss OPC Data Access Specification, **) nicht verwendet/verfügbar bei OPC

r="Read from Server" w="Write to Server"

- *Definition*

Read from Server (vom Server lesen) bzw. Write to Server (zum Server schreiben) definieren die Datenpunktbezeichnung (Adresse oder Name) im Server für eine Subskription <**CX**> und/oder <**SX**> oder für eine Abfrage <**QX**>. Die Eingaben bestimmen, ob ein Datenpunkt über die entsprechende Verbindung nur gelesen (Read only), nur geschrieben (Write only) oder gelesen UND geschrieben (Read/Write) werden soll. Gleichzeitig ermöglichen die Einträge eine Umbenennung (Renaming) der Datenpunkte zwischen Client und Server. Die Eingaben dürfen nur zusammen mit der lokalen Adresse **a**=".." - oder dem Netzwerk Namen **n**=".." des Clients angegeben werden:

entweder Adressen ...

```
<CX> und/oder
<SX> oder
<QX> (nur in Telegramm)
   <P a="[Addr|Wildcard]" r="[Addr|Wildcard|=]" [w="[Addr|Wildcard|=]"]/>
```

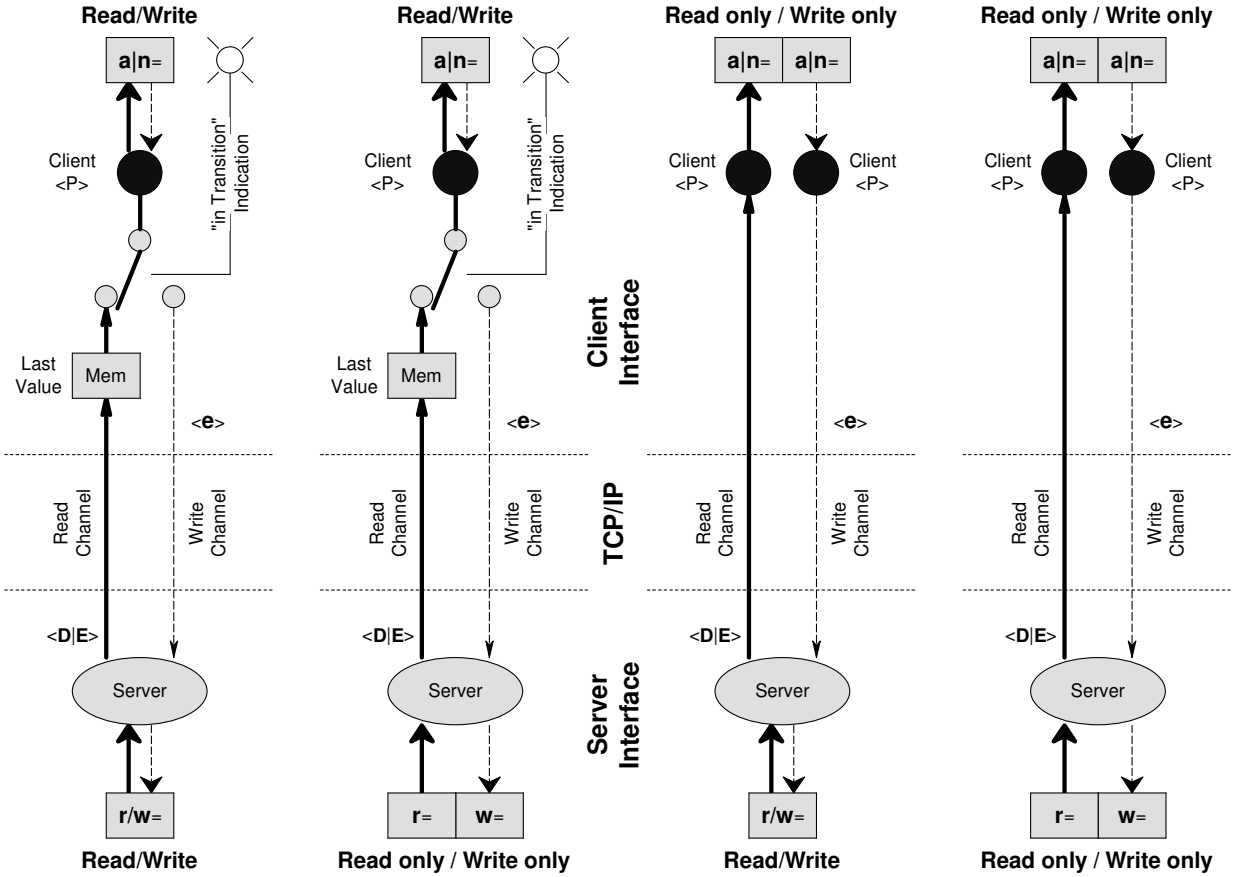
oder Namen ...

```
<CX> und/oder
<SX> oder
<QX> (nur in Telegramm)
   <P n="[Name|Wildcard]" r="[Name|Wildcard|=]" [w="[Name|Wildcard|=]"]/>
```

Read from Server **r**=".." und Write to Server **w**=".." können Wildcards (*?) enthalten. Diese beziehen sich auf die entsprechenden Stellen der Bezeichnung **a**|**n**=".." im Client und sind an beliebiger Stelle, jedoch in gleicher Reihenfolge im Ausdruck **r**=".." bzw. **w**=".." für den Server einzufügen. Die Anzahl der Wildcards in **a**|**n**=".." und **r**=".." bzw. **w**=".." muss übereinstimmen. Ein Gleichheitszeichen **r**="=" bzw. **w**="=" bedeutet, dass die Bezeichnung im Server gleich ist wie diejenige im Client. Die Attribute ermöglichen mehrere Definitionen gleichzeitig:

- Die Übertragungsrichtung Lesen (Read) und/oder Schreiben (Write) kann bestimmt werden. Der bi-direktionale Datenaustausch (Read/Write) erfordert beide Attribute **r**=".." und **w**="..", wird nur **r**=".." eingetragen, dann definiert das eine Read Only - bzw. bei nur **w**=".." eine Write Only Übertragung. Ausschliessliche Write Only Definitionen sind nur dann sinnvoll, wenn der tatsächliche Wert des Servers im Client nie zur Verfügung stehen soll.
- Bi-direktional definierte Übertragungen werden automatisch mit einer Transitionsanzeige **q**="gT" begleitet, wenn der Wert im Client verändert wurde. Der lokale DxNode spiegelt eine Eingabe im Client sofort ("Action=Reaction" statt ⌛-Anzeige) zusammen mit der Transition, während der Befehl im Hintergrund an den Server weitergeleitet - und zurückgemeldet wird, ➲ siehe auch **upd_delay**="Update Delay".
- Die Datenpunkte können in Client **a**|**n**=".." und Server **r**="..", **w**=".." unterschiedlich bezeichnet werden d.h. Datenpunkte können mit der Subskription umbenannt werden (Renaming).
- Ein Datenpunkt **a**|**n**=".." im Client kann zwei Datenpunkte **r**=".." und **w**=".." im Server haben. Man kann damit das Netzwerk auf einen Namen pro Datenpunkt standardisieren und für individuelle Anbindungen nach Bedarf separate Adressen für Read und Write definieren.
- Zwei Datenpunkte **a**|**n**=".." , z.B. ein Read Only und ein Write Only im Client können auch den gleichen Datenpunkt **r**=".." bzw. **w**=".." im Server haben. Damit ist ein pseudo bi-direktionaler Zugriff auf den gleichen Datenpunkt im Server definierbar, was der herkömmlichen Methode gängiger Systeme wie z.B. OPC entspricht. Diese Methode bietet im Gegensatz zur oben beschriebenen bi-direktionalen Übertragung keine Transitionsanzeige mit Action=Reaction und benötigt zwei Datenpunkte im Client.

Die nachfolgende Darstellung zeigt die Übertragung der Elementdaten <**D**|**E**|**e**> für Read und Write beim bi-direktionalen bzw. beim uni-direktionalen Datenaustausch. In beiden Fällen können für den Server nur ein Datenpunkt (bi-direktional) oder deren zwei für Lesen und Schreiben definiert werden:



Wichtig! beim (Re)Initialisieren der Verbindung werden nur die Read Daten (fette Linien) mit <**D**> im Client abgeglichen (synchronisiert) d.h. Write Daten <**e**> (gestrichelt) werden beim Initialisieren nicht übermittelt und können auch nicht gespeichert und weitergeleitet werden (kein Store&Fwd).

- *Anwendung*

Drei Beispiele sollen die wichtigsten Möglichkeiten erläutern:

a) Wildcards (*?) ermöglichen die Umbenennung ganzer Gruppen von Datenpunkten. Z.B. bedeutet der Ausdruck <**P a**|**n**="A*B*C*" **r**="rdA*B*C*", **w**="wrA*B*C*" />, dass ein Datenpunkt im Client mit Name "Ax_By_Cz" vom Server mit Name "rdAx_By_Cz" gelesen - und mit Name "wrAx_By_Cz" an den Server übermittelt würde. Das Beispiel zeigt auch, dass EIN Datenpunkt im Client ZWEI getrennte Datenpunkte im Server haben kann, einen zum Lesen und einen zum Schreiben.

b) Read from Server **r**=".." und Write to Server **w**=".." können mit einem Gleichheitszeichen "=" definiert werden. Dies bedeutet, dass die Adresse bzw. der Name **a**|**n**=".." des Clients 1:1 für den Server übernommen werden soll. Der Ausdruck <**P a**|**n**="A*B*C*" **r**="=", **w**="=" /> bedeutet, dass die Bezeichnungen in Client und Server gleich sind und der gleiche Datenpunkt im Server sowohl gelesen, wie auch geschrieben wird.

c) Andererseits ist der Ausdruck <**P a**|**n**="A*B*C*" **r**="=", **w**="wrA*B*C*" /> eine Kombination von a) und b) indem der zu lesende Datenpunkt in Client und Server zwar gleich heissen, jedoch im Server auf einen separaten Datenpunkt geschrieben wird.

Die Methoden b) oder c) sind für den bi-direktionalen Datenaustausch (Read/Write) empfohlen. Man hat dann im Netzwerk immer nur einen Datenpunkt, kann aber je nach Bedarf für die Anbindung zwei verschiedene Adressen für Read und Write definieren.

rd_interval="Read Interval" wr_interval="Write Interval"

- *Definition*

Lese- und Schreibintervall definieren eine maximale Wartezeit in [ms] zwischen zwei nacheinander gesendeten Telegrammen <**X0**><**P**> mit Elementdaten <**D**|**E**> (Read from Server) bzw. <**e**> (Write to Server) für Verbindungen mit einer Subskription <**CX**> oder <**SX**> (Dauerauftrag). Die Parameter wirken nur auf den echten Datenaustausch, wobei jedes Telegramm eine nicht eingeschränkte Anzahl von Datenpunkten und Elementdaten <**P** <**E**|**e** ... /></**P**> enthalten kann, andere Telegramme z.B. zur Subskription unterliegen nicht einem Intervall d.h. sie werden spontan gesendet. Andererseits bewirkt ein Alive-Telegramm, dass die Wartezeit abgelaufen ist, weil ja dann eine Übermittlung zur Verbindungsüberwachung stattfinden muss. Die Intervalle sind also sinnvollerweise kürzer zu halten als die Lebenszeichen-Überwachungszeit **alive**=".." für die entsprechende Verbindung.

Das Intervall steuert den Datenfluss indem Ereignisse vor dem Versenden gesammelt und gemeinsam mit EINEM Telegramm übermittelt werden. Dies ermöglicht eine optimierbare Datenübertragungsrate, weil ja nicht jedes Ereignis unverzüglich übermittelt werden muss und damit eine ausgeglichenere Systembelastung erreicht werden kann.

Die Intervallzeiten in [ms] können global für alle Verbindungen im Element <**Node**> - oder individuell pro Verbindung im Element <**Connect**> eingestellt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **rd_interval**="[0..65535]" **wr_interval**="[0..65535]" **upd_delay**=".." [..]/>

und/oder ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **rd_interval**="[0..65535]" **wr_interval**="[0..65535]" **upd_delay**=".." [..]/>

Die Standardwerte sind **rd_interval**="200" und **wr_interval**="200" [ms], dies bedeutet, dass Elementdaten normalerweise spätestens nach 200 [ms] übermittelt werden. Der Standardwert wird durch den Eintrag in <**Node**> ersetzt. Wenn ein Eintrag in <**Connect**> vorhanden ist, dominiert dieser für die entsprechende Verbindung.

- *Anwendung*

Die Intervalleinstellung stellt auch eine bewusst erzwungene "Übertragungsverzögerung" dar, damit Ereignisse, die während der Wartezeit eintreten, effizient gepackt als Sammeltelegramme übermittelt werden können. Tatsächlich wird damit die Reaktionszeit zwischen Bedieneingabe und Rückmeldung etwas erhöht, was aber mit der Transitionsanzeige **q**="gT" während der Update-Verzögerung **upd_delay**=".." für bi-direktionale Datenpunkte kein Problem darstellt: der lokale DxNode spiegelt die Eingabe im Client sofort mit Transition **q**="gT" ("Action=Reaction" statt ⌛-Anzeige) während der Befehl im Hintergrund an den Server weitergeleitet - und zurückgemeldet werden kann.

Die Methode erlaubt einen hohen Bedienkomfort bei gleichzeitiger Entlastung der Kommunikation, weil Befehle und Antworten nicht möglichst schnell, sondern effizient gepackt übertragen werden können. Tatsächlich unterstützt die Methode sogar sehr "langsame Verbindungen" mit komfortabler Bedienung - man sieht dann einfach eine etwas länger dauernde Transitionsanzeige (➲ siehe auch **upd_delay**="..").
Andererseits erlaubt die Methode eine Optimierung des Datendurchsatzes ohne Komforteinbusse bei der Bedienung, d.h. die an sich widersprüchliche Anforderung einer schnellen Bedienreaktion bei hohem Datenaufkommen ist gelöst und kann vom Anwender genutzt werden.

Lese- und Schreibintervall, zusammen mit der maximalen Telegrammlänge **max_telegr_length**=".." definieren eine maximale Übertragungsrate pro Verbindung. Das Leseintervall bestimmt auch eine regelmässige Aktualisierung des Prozessabbildes im Client, wenn Daten im Server geändert wurden. Mit dem Schreibintervall werden schnell veränderte Daten, wie sie etwa mit der Auto-Repeat-Funktion einer Tastatur verursacht werden können, gepackt übermittelt. Eine Anwendung kann so z.B. nur den Letztwert verarbeiten und muss nicht jeden Inkrementalwert einzeln auswerten.

☞ Das Leseintervall kann für jede Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].value.**rd_interval** abgefragt werden.

☞ Das Schreibintervall kann für jede Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].value.**wr_interval** abgefragt werden.

reconnect_cycle="Reconnect Cycle"

- *Definition und Anwendung*

Zeitintervall zum Wiederherstellen der Verbindung nach einem Unterbruch bzw. nach einer Umschaltung in [sec]. Die Zeit kann global für alle Verbindungen im Element <**Node**> - oder individuell pro Verbindung im Element <**Connect**> eingestellt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **reconnect_cycle**="[1..65535]" [..]/>

und/oder ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **reconnect_cycle**="[1..65535]" [..]/>

Der Standardwert ist **reconnect_cycle**="1" [sec], dies bedeutet, dass DxNode versucht, eine unterbrochene bzw. umgeschaltete Verbindung im Sekundentakt wieder herzustellen.

Das Zeitintervall kann für jede Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].cmdio.**reconnect_cycle** abgefragt oder verändert werden. Die parametrierte Vorgabe **reconnect_cycle**=".." wird dabei nicht verändert.

rid="Receiver ID" siehe nid=".."

- *Definition und Anwendung*

In Telegrammen <**X0**> kann eine Sender und Empfänger Identifikation als **sid**=".." und **rid**=".." übermittelt werden. Der Inhalt wird in Element <**Node**> des Senders und Empfängers mit dem Eintrag für die Knoten-Identifikation **nid**=".." bestimmt. Für gültige Eingaben ➲ siehe **nid**="Node ID".

s="Status and Bit Filter" siehe auch v=".."

- *Definition und Anwendung*

Der User Status stellt eine anwendungsseitig definierbare Qualität als 8 Bit Unsigned Integer dar. Der Status entspricht der OPC Vendor Specific Quality und bezieht sich auf den Elementwert **v**="..". Er kann bei Bedarf in der <**DPList**> oder in den Steuerelementen <**Link**...> pro Datenpunkt-Auswahl <**P**> als <**D**|**E**|**e s**=".." /> vorgegeben werden. Gültige Einträge sind:

<**D**|**E**|**e** [**v**=".."] **s**="[0..255]" [..]/>

Der Standardwert ist **s**="0", d.h. kein Status Bit gesetzt.

☞ Es werden immer alle 8 Bits als Statuswert 0..255 übertragen.
Einzelne Bits können nur in Anwendungen encodiert oder decodiert werden.

Ein Statuswert kann in Binärform als Filter pro Subskription <**CX**>, <**SX**> oder Abfrage <**QX**> und/oder deren untergeordneter Datenpunkte-Auswahl <**P**> definiert werden. Gültige Einträge sind:

<**CX** [**s**="[bbbbbbbb]"][..]> und/oder
<**SX** [**s**="[bbbbbbbb]"][..]> oder
<**QX** [**s**="[bbbbbbbb]"][..]>
  <**P a**=".." **r**=".." [**w**=".."][**s**="[bbbbbbbb]"][..]/> oder
  <**P n**=".." **r**=".." [**w**=".."][**s**="[bbbbbbbb]"][..]/>

... wobei [bbbbbbbb] das Filter als Binärausdruck darstellt
  | |
  | least significant bit (Wertigkeit 1)
  most significant bit (Wertigkeit 128)

... und "b" für alle 8 Status Bits zu definieren ist:

b=? Status Bit kann 0 oder 1 sein oder
b=0 Status Bit muss 0 sein oder
b=1 Status Bit muss 1 sein, damit die Meldung übermittelt wird.

Der Standardwert ist **s**="????????" (kein Filter), d.h. die Meldung wird immer übertragen. Beispiele:

<**CX s**="10??????"> ← es werden nur Datenpunkte mit **s**="[128...191]" übermittelt
<**SX**><**P n**=".." **r**="=" **s**="???????1"/></**SX**> ← es werden nur Datenpunkte mit **s**="[odd]" übermittelt

☞ Gesperrte Datenpunkte werden bei Verbindungsherstellung NICHT übertragen.
D.h. wenn ein Wert zur Zeit der Synchronisation gesperrt war, wird er erst wieder mit der nächsten Änderung NACH Aufhebung der Sperrung nachgeführt.

sid="Sender ID" siehe nid=".."

- *Definition und Anwendung*

In Telegrammen <**X0**> kann eine Sender und Empfänger Identifikation als **sid**=".." und **rid**=".." übermittelt werden. Der Inhalt wird in Element <**Node**> des Senders und Empfängers mit dem Eintrag für die Knoten-Identifikation **nid**=".." bestimmt. Für gültige Eingaben ➲ siehe **nid**="Node ID".

store_fwd_buffer="Store&Fwd Buffer"

- *Definition und Anwendung*

Die Store&Fwd Buffer Grösse in [KB] kann für alle Verbindungen im Element <**Node**> global - oder individuell pro Verbindung im Element <**Connect**> eingestellt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **store_fwd_buffer**="[0..100000]" [..]/>

und/oder ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **store_fwd_buffer**="[0..100000]" [..]/>

Der Standardwert ist **store_fwd_buffer**="1000" [KB], dies bedeutet, dass pro Verbindung <**Connect**> mit Subskription <**SX**> und Store&Fwd Parametrierung **attr**="S" ein Puffer von 1000 KB eröffnet wird. Der Standardwert wird durch den Eintrag in <**Node**> ersetzt. Wenn ein Eintrag in <**Connect**> vorhanden ist, dominiert dieser für die entsprechende Verbindung.

Je nach Länge der Datenpunktbezeichnung **n**=".." und der übertragenen Attribute gemäss **attr**=".." wird eine gespeicherte Meldung schätzungsweise 0.2..0.5 [KB] benötigen d.h. es können mit dem Standardwert ca. 2000..5000 Meldungen pro Verbindung gespeichert werden.

t="Timestamp" siehe auch v=".."

- *Definition und Anwendung*

Zeitstempel dienen der zeitlichen Zuordnung von Meldungen und werden für Telegramme <**X0**> bzw. <**X0**><**P**> für deren Elementdaten <**D**|**E**|**e**> vom Sender vergeben. Fehlende Zeitstempel werden von DxNode automatisch mit der lokalen Empfangszeit ergänzt, d.h. eine externe Anwendung muss die Daten nicht zeitstempeln. Gültige Darstellungen sind:

<**X0** [**l**=".."] **t**="[YYYY-MM-DDThh:mm:ss.xxx[±hh:mm]]" [..]>

und/oder ...

<**P a**|**n**="."><**D**|**E**|**e** [**v**=".."] **t**="[YYYY-MM-DDThh:mm:ss.xxx]" [..]/></**P**>

... wobei YYYY=Jahr, MM=Monat, DD=Tag, hh=Stunden, mm=Minuten, ss=Sekunden und xxx=ms

Die Zeitstempel sind im dateTime Format (ISO-8601 entspricht **f**="%yh") mit einer fixen Auflösung von [ms] d.h. 23 Zeichen dargestellt. Die Zeit ist Universal Time Coordinated (UTC), also weltweit eindeutig, ohne Zeitzonen- oder Sommer/Winterzeitverschiebung. Bei Bedarf kann ggf. die Zeitzone im Telegramm <**X0**> als [±hh:mm] angehängt werden. Dies ermöglicht zwar die Erkennung der Systemeinstellung im Sender und damit die Berechnung der lokalen Sendezeit, hat aber keinen Einfluss auf die Verarbeitung im Empfänger, da der Zeitstempel als UTC definiert ist. Der Standardwert ist **t**="[Nullstring]".

Der Zeitstempel des zuletzt empfangenen Telegrammes <**X0**> kann pro Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].value.last_rcv abgefragt werden.

Der Zeitstempel des zuletzt gesendeten Telegrammes <**X0**> kann pro Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].value.last_snd abgefragt werden.

Der Zeitstempel in Elementdaten <**D**|**E**|**e**> bezieht sich ausschliesslich auf den Elementwert **v**=".." und kann bei Bedarf in der <**DPList**> oder in den Steuerelementen <**Link...**> pro Datenpunkt-Auswahl <**P**> als <**D**|**E t**=".." /> voreingestellt werden. Zeitstempel in Elementdaten <**D**|**E**|**e**> werden von DxNode geprüft und ggf. korrigiert, sollten sie einen logisch festgelegten Bereich überschreiten. Eine allfällige Zeitkorrektur wird dann mit **tc**=".." festgehalten, ➲ siehe **v**="Value".

tc="Timestamp Corrective" siehe auch v=".."

- *Definition und Anwendung*

Eine allfällige Zeitstempelkorrektur in [ms] bezieht sich auf den Zeitstempel **t**=".." des Wertes **v**="..". DxNode korrigiert ungültige, mit Telegrammen <**X0**> übermittelte Zeitstempel <**D**|**E**|**e t**=".." /> automatisch und fügt die berechnete Zeitkorrektur den Elementdaten <**D**|**E**|**e**> als **tc**="[NonZero]" an. Gültige Darstellungen sind:

<**D**|**E**|**e** [**v**=".."] **t**=".." **tc**="[-2147483648...+2147483647]" [..]/>

Das entspricht ca. ±596 [h] oder ±24 [Tg], der Standardwert ist **tc**="0", dieser Wert wird im Telegramm nicht übertragen, d.h. der Empfänger bzw. DxNode muss den Wert automatisch **tc**="0" setzen. DxNode bewertet einen Zeitstempel der Elementdaten <**D**|**E t**=".." [..]/> in einem Telegrammen <**X0**><**P**> als "ungültig" wenn ...

- **t**="[Timestamp]" **kleiner** (älter) ist als der in DxNode bereits - bzw. zuletzt gespeicherte Zeitstempel, dann wird **t**="[LastTimestamp]" und **tc**="[Timestamp-LastTimestamp]" d.h. negativ
- **t**="[Timestamp]" **grösser** (jünger) ist als die um die Zeittoleranz <**Node tt**=".." /> erhöhte, aktuelle Knotenzeit, dann wird **t**="[ActualNodeTime]" und **tc**="[Timestamp-ActualNodeTime]" d.h. positiv

In beiden Fällen kann die Originalzeit des Partners mit **t**="[Timestamp]+[TimeCorrective]" berechnet werden. Wenn die Differenz den zulässigen Bereich überschreitet, wird der Endbereichswert als **tc**=".." gespeichert d.h. wenn **tc**="[-2147483648|+2147483647]", ist der ursprüngliche Zeitstempel nicht mehr berechenbar - er war dann ursprünglich älter bzw. jünger als 24 Tage.

Eine Zeitstempelkorrektur **tc**="[NonZero]" wird wie andere Elementdaten <**D**|**E**|**e**> behandelt und immer automatisch zusammen mit **t**=".." in den Telegrammen <**X0**><**P**> eingefügt d.h. an alle beteiligten Knoten weitergeleitet. Die Korrektur ist also in allen Knoten erkennbar und der ursprüngliche Zeitstempel kann bis ± 24 Tage Differenz exakt ermittelt werden.

tgt="Time Greater Than" tlt="Time Less Than"

- *Definition und Anwendung*

Eine Zeiteinschränkung grösser/kleiner bzw. von../bis.. kann für Subskriptionen <**CX**> oder <**SX**> (Daueraufträge), für einen Single Request <**QX**> (Einzelabfrage) und/oder deren untergeordneter Datenpunkte-Auswahl <**P**>, sowie für einen History Request <**HX**> (Historieabfrage) definiert werden. Gültige Einträge sind:

<**CX** [**tgt**="[dateTime|time]"][**tlt**="[dateTime|time]"][..]> und/oder
<**SX** [**tgt**="[dateTime|time]"][**tlt**="[dateTime|time]"][..]> oder
<**QX** [**tgt**="[dateTime|time]"][**tlt**="[dateTime|time]"][..]> (nur in Telegramm)
  <**P a**=".." **r**=".." [**w**=".."][**tgt**="[dateTime|time]"][**tlt**="[dateTime|time]"][..]/> oder
  <**P n**=".." **r**=".." [**w**=".."][**tgt**="[dateTime|time]"][**tlt**="[dateTime|time]"][..]/>

oder ...

<**HX** [**tgt**="[dateTime|time]"][**tlt**="[dateTime|time]"]> (nur in Telegramm)

... wobei [dateTime|time] einen Zeitpunkt im Format **f**="%yh" (YYYY-MM-DDThh:mm:ss.xxx) darstellt. Die Datum und Zeitangaben müssen Universal Time Coordinated (UTC) sein, d.h. weltweit eindeutig, ohne Zeitzonenverschiebung oder Sommer/Winterzeitumschaltung. Speziell ist, dass Zahlen von links mit Wildcards (?) ersetzt - und alle Zeichen von rechts weggelassen werden dürfen. Zum Beispiel wird "2008" als "2008-01-01T00:00:00.000" interpretiert. Der Ausdruck "????-??-??T15:27" wird als "Datum des aktuell zu vergleichenden Zeitstempels mit der Zeit T15:27:00.000" verstanden. Bei fehlendem Attribut, wird keine Prüfung vorgenommen.

Die Zeiteinschränkung (Filter) bezieht sich für die entsprechende Subskription jeweils auf Lesen und Schreiben und bedeutet, dass nur Ereignisse in Telegrammen <**X0**><**P**> übermittelt werden, deren Zeitstempel <**D**|**E**|**e t**=".." /> innerhalb des Zeitfensters **tgt**=".." (grösser) und **tlt**=".." (kleiner) liegen. Bei Eingabe von Wildcards (?) werden die entsprechenden Stellen durch die Werte des aktuell zu vergleichenden Zeitstempels ersetzt und damit quasi ignoriert. Das Filter wirkt dann periodisch, fortlaufend z.B. jeden Tag d.h. es beginnt und endet immer automatisch um Mitternacht 00:00:00, sofern nicht anders definiert. Zum Beispiel bedeuten ...

<**CX tgt**="2009-12" **tlt**="2010-12"> ← von Anfang Dez. 2009 bis Ende Nov. 2010 d.h. 1 Jahr lang
<**CX tgt**="????-??-15T20:44"> ← ab 15. jeden Monats von 20:44h bis jeweils Ende des Monats
<**CX tgt**="????-??-??T09:30" **tlt**="????-??-??T17:45"> ← jeden Tag von 09:30h bis 17:44:59.999h
<**CX tlt**="????-??-??T15:00:02"> ← jeden Tag von 00:00h bis jeweils 15:01:59.999h
<**CX tgt**="2008-07-15" **tlt**=????-??-??T"00:10"> ← ab 15. Juli 2008 täglich 00:00h bis 00:09:59.999h

☞ Gesperrte Datenpunkte werden bei Verbindungsherstellung NICHT übertragen.
D.h. wenn ein Wert zur Zeit der Synchronisation gesperrt war, wird er erst wieder mit der nächsten Änderung NACH Aufhebung der Sperrung nachgeführt.

Die Zeiteinschränkung wird auch bei der Herstellung eines <**Connect**> mit Parametrierung **attr**="S" für Store&Fwd im Server <**SX**> verwendet. Dabei ist die Zeiteinschränkung gegeben durch den Zeitstempel des zuletzt empfangenen Telegrammes <**X0**> im Client für diese Verbindung. Dieser Zeitstempel kann über den ➲ internen Datenpunkt [Prefix][ConnectName].value.last_rcv abgefragt werden. Gültige Darstellungen sind:

<**X0**>
<**Connect cn**="[ConnectName]"><**Switch** [**cn**=".."][**tgt**="[value.last_rcv]"]/></**Connect**>

Dies bedeutet, dass nur Telegramme mit Zeitstempel <**X0 t**=".."> grösser als der Zeitstempel des zuletzt empfangenen Telegrammes [value.last_rcv] übertragen werden sollen. Der Befehl <**Switch**/> (switch to named connect) fordert die Vermittlung mit der parametrierten Verbindung **cn**=".." an.

Falls die Serverseite einen Store&Fwd Buffer unterstützt, werden zuerst die aufgezeichneten Telegramme <**X0 t**=".."> grösser als [value.last_rcv] nachsynchronisiert, d.h. die Ereignisse für eine unterbrochene Verbindung würden nahtlos, ohne Überlappung nachgeführt. Anschliessend erfolgt die normale Synchronisation aller aktuellen Daten.

Bei einer Umschaltung auf einen anderen Server z.B. im Redundanzfall, ist nicht gewährleistet, dass der Zeitstempel [value.last_rcv] kleiner ist als alle noch nicht übertragenen Telegramme aus dem Store&Fwd Buffer. Daher ist eine im Server festgelegte Zeittoleranz **tt**=".." als Überlappungszeit zu gewähren, damit keine Ereignislücke entsteht. Die Zeittoleranz **tt**=".." ist gegeben durch die maximal mögliche Differenz zweier Zeitstempel für ein und dasselbe Ereignis in den zwei beteiligten Rechnersystemen. Der Zeitstempel [value.last_rcv] wird im Server um die Zeittoleranz **tt**=".." reduziert womit ggf. gewisse Ereignisse doppelt nachgeführt werden. Andererseits ist nicht zu vermeiden, dass auch die Zeitstempel der Elementdaten <**D**|**E**|**e t**=".." /> zweier Systeme unterschiedlich sind und bei Empfang mit einer Zeitstempelkorrektur **tc**=".." versehen werden müssen.

Für nicht redundante Server könnte man die Zeittoleranz **tt**="0" einstellen womit Ereignisse wie oben beschrieben ohne Überlappung nachgeführt werden. Andererseits ist es durchaus sinnvoll, generell eine gewisse Zeittoleranz **tt**=".." zuzulassen, weil ja die Zeitstempel nie 100% synchronisiert sind und mit einer Zeittoleranz **tt**=".." auch unnötige Zeitstempelkorrekturen **tc**=".." verhindert werden.

tlt="Time Less Than" siehe tgt=".."

- *Definition und Anwendung*

Eine Zeiteinschränkung grösser/kleiner bzw. von../bis.. kann pro Subskription <**CX**|**SX**> oder Anfrage <**QX**|**HX**> und/oder deren untergeordneter Datenpunkte-Auswahl <**P**> definiert werden. Für gültige Eingaben und Anwendung ➲ siehe **tgt**="Time Greater Than.

to="Target File" siehe from=".."

- *Definition und Anwendung*

Eine Quellen- und eine Zieldatei sind im Download-Befehl <**GetFile**> anzugeben. Für gültige Eingaben und Anwendung ➲ siehe **from**="Source File".

tt="Time Tolerance"

- *Definition und Anwendung*

Die Zeittoleranz in [ms] ist gegeben durch die mögliche Differenz zweier Zeitstempel für ein und dasselbe Ereignis in den beteiligten Rechnersystemen. Die Zeittoleranz erlaubt den Vergleich von Zeitstempeln verschiedener Systeme in vorgegebenen Grenzen. Um überhaupt Zeitstempel vergleichen zu können, ist es unabdingbar, dass die Systemzeit der Rechner über das Netzwerk abgeglichen wird. Allerdings ist es nicht möglich, von zwei oder mehreren Systemen gestempelte Zeiten absolut, z.B. auf die [ms] genau synchron zu halten. Aus diesem Grunde kann pro DxNode die Zeittoleranz im Element <**Node**> für alle Zeitprüfungen festgelegt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **tt**="[0..65535]" [..]/>

Der Standardwert ist **tt**="300" [ms], dies bedeutet, dass ein gegenüber dem lokalen System vorauseilender Zeitstempel um 300 [ms] nicht als Fehler betrachtet und daher nicht gemäss Zeitstempelkorrektur **tc**=".." korrigiert werden muss.

Die Zeittoleranz **tt**=".." wird auch verwendet um bei einer Umschaltung auf einen anderen Server z.B. im Redundanzfall, eine gewisse Überlappungszeit zu gewähren damit alle im Toleranzbereich zu übertragenden Telegramme aus dem Store&Fwd Buffer nachgeführt werden können. Dabei wird der Zeitstempel des zuletzt empfangenen Telegrammes im Client bei Verbindungs(wieder)herstellung als **tgt**=[value.last_rcv] an den Server übermittelt und dort um die Zeittoleranz **tt**=".." reduziert, womit der Store&Fwd Buffer ab diesem (um die Zeittoleranz **tt**=".." reduzierten) Zeitpunkt im Client nachgeführt werden kann.

u="Engineering Unit" siehe auch v=".."

- *Definition und Anwendung*

Die Masseinheit bezieht sich auf den Wert **v**=".." und kann bei Bedarf in der <**DPList**> oder in den Steuerelementen <**Link...**> pro Datenpunkt-Auswahl <**P**> als <**D**|**E**|**e u**=".." /> vorgegeben werden. Gültige Einträge sind:

<**D**|**E**|**e** [**v**=".."] **u**="[-|1|1/s|%|%RF|°C|km|m|mm|km/h|m3/h|RPM|ppm|Pa|kPa|cd|h|min|s|ms|..]"/>

Der Standardwert ist **u**="[Nullstring]". Eine Liste der zulässigen Masseinheiten kann als XML Schema für die Parametrierung definiert werden, zum Beispiel (Liste nicht abschliessend):

Unit	Beschreibung
u="-"	Keine Einheit
u="1"	Einheit "1" (Normierter Wert)
u="1/s"	Drehzahl
u="%"	Prozent
u="%RF"	Prozent Relative Feuchtigkeit
u="°C"	Grad Celsius
u="°K"	Grad Kelvin
u="A"	Ampère
u="mA"	Milliampère
u="Bq"	Becquerel
u="Byte"	Byte
u="KB"	Kilobyte
u="MB"	Megabyte
u="F"	Farad
u="pF"	Picofarad

Unit	Beschreibung
u="lm"	Lumen
u="lx"	Lux
u="m"	Meter
u="m/s"	Meter pro Sekunde
u="mm"	Millimeter
u="m2"	Quadratmeter
u="m3"	Kubikmeter
u="m3/h"	Kubikmeter pro Stunde
u="km"	Kilometer
u="km/h"	Kilometer pro Stunde
u="mol"	Mol
u="N"	Newton
u="Nm"	Newton Meter
u="Ohm"	Ohmscher Widerstand
u="pH"	pH-Wert

Unit	Beschreibung
u="Hz"	Hertz
u="kHz"	Kilohertz
u="MHz"	Megahertz
u="J"	Joule
u="J/m3"	Joule pro Kubikmeter
u="g"	Gramm
u="kg"	Kilogramm
u="g/m3"	Gramm pro Quadratmeter
u="mg/m3"	Milligramm pro Quadratmeter
u="t"	Tonne
u="Pa"	Pascal
u="kPa"	Kilo Pascal
u="cd"	Candela

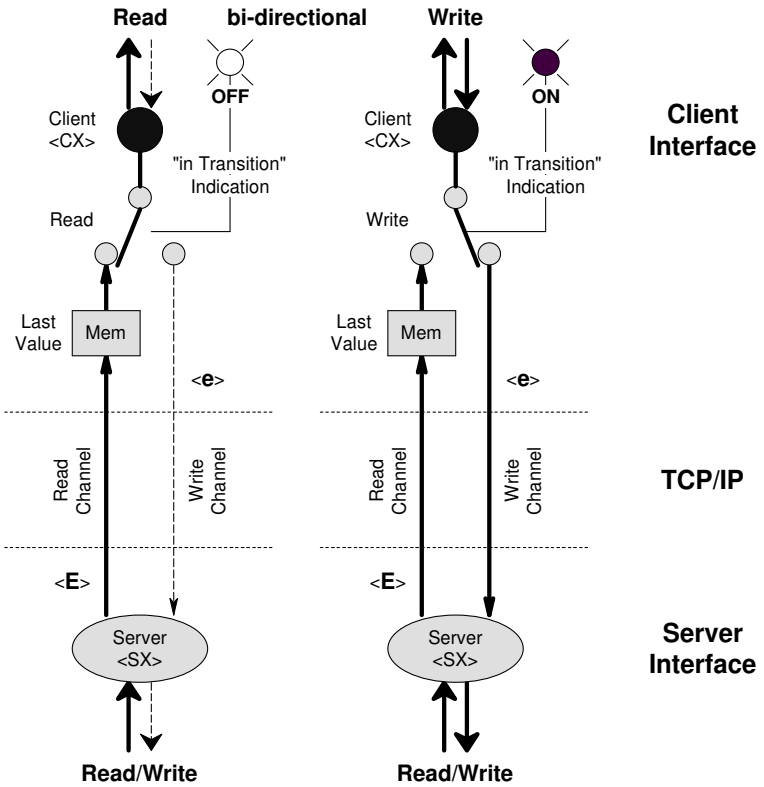
Unit	Beschreibung
u="ppm"	Parts per Million
u="RPM"	Umdrehungen pro Minute
u="s"	Sekunden
u="ms"	Millisekunden
u="h"	Stunden
u="min"	Minuten
u="W"	Watt
u="kW"	Kilowatt
u="MW"	Megawatt
u="kWh"	Kilowattstunden
u="MWh"	Megawattstunden
u="W/m2"	Watt pro Quadratmeter

upd_delay="Update Delay" fast_update="Control Flag"

- *Definition*

Die Update-Verzögerung in [ms] ist die Zeit, die ein Client zulässt, um einen Schreibbefehl <**e**> (Write/Advise) vom Server zurückzulesen <**D**|**E**> (Read). Die Verzögerungszeit entspricht dabei einer Befehlslaufzeitüberwachung, die automatisch für bi-direktional deklarierte Datenpunkte (Read/Write) im Client <**CX**><**P nIa**=".." **r**=".." **w**=".." /></**CX**> wirksam wird, wenn Elementdaten <**e**> eines solchen Datenpunktes im selben Client verändert werden. Während der Verzögerungszeit wird eine zufällige Aktualisierung der Elementdaten <**D**|**E**> aus dem Server blockiert und so ein unerwünschtes Flattern der Eingabe während der Bedienung verhindert, d.h. der Bediener hat während der Update-Verzögerung scheinbar die Führung über das Signal im lokalen System, natürlich nur, wenn er vorgängig die entsprechende Bedienberechtigung hatte.

Die Update-Verzögerung wirkt im Client wie ein Schalter, der normalerweise auf "Read" steht und nur während der Befehlseingabe auf "Write" gestellt wird, bis der Befehl vom Server bestätigt ist und der Schalter - nach Ablauf der Übertragung - wieder auf "Read" gestellt werden kann:



Die aktive Update-Verzögerung wird pro Datenpunkt mit Quality **q**="gT" (in Transition) angezeigt, d.h. bei entsprechender Animation sieht der Bediener, dass der (Write) Befehl "unterwegs" ist, bis er vom Server bestätigt wird. Nach Ablauf der Verzögerungszeit wird automatisch der aktuelle, vom Server bestimmte (Read) Zustand angezeigt, wobei der Server grundsätzlich jeden Zustand übertragen darf. Bei Ausbleiben der Bestätigung z.B. bei Verbindungsunterbruch oder wenn der Server den Befehl nicht quittiert, fällt der Zustand automatisch wieder auf den letzten, vor der Bedienung gespeicherten Wert (Mem) zurück. Generell gilt der zeitlich zuletzt veränderte Wert im Server als "aktuell".

Die Update-Verzögerung muss nur die Übertragungs- und Reaktionszeit bis zum ersten, direkt verbundenen Server im Netzverbund abdecken, weil ein Befehl immer zusammen mit der Transition **q**="gT" weitergeleitet und quittiert wird, bis der letzte, zuständige Server in der Kette die Transition aufhebt, d.h. den aktuellen Befehls-Zustand mit **q**="[anyExcept gT]" zurückmeldet.

Die Update-Verzögerung in [ms] kann global für alle Verbindungen im Element <**Node**> - oder individuell pro Verbindung im Element <**Connect**> eingestellt werden. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **upd_delay**="[0..65535]" [**rd_interval**=".."][**wr_interval**=".."][..]>

und/oder ...

<**DxCnfLoc**> oder <**DxCnfExt**>
  <**Connect cn**=".." **upd_delay**="[0..65535]" [**rd_interval**=".."][**wr_interval**=".."][..]>

Der Standardwert ist upd_delay="3000" [ms], dies bedeutet, dass die Transitionsanzeige **q**="gT" bei ausgeschaltetem **fast_update**="false" immer für 3 sec ansteht.

Generell muss die Update-Verzögerung grösser sein als die Summe des für die Verbindung definierten Leseintervalls **rd_interval**=".." plus das Schreibintervall **wr_interval**="..", weil diese Werte bewusst als "Übertragungsverzögerung" eingestellt werden (➲ siehe **rd_interval**=".."). Der minimale Wert sei als Faustregel wie folgt vorgegeben:

upd_delay = 2 * Netzübertragungszeit + **rd_interval** + **wr_interval** + Server Reaktionszeit

Dieser Wert darf aber z.B. wegen nicht deterministischen Verhaltens problemlos höher eingestellt werden. Der Bediener sieht dann zwar eine etwas länger andauernde Transitionsanzeige aber er muss nie warten, bis er sieht was er eingegeben hat: der lokale DxNode spiegelt die Eingabe im Client sofort mit Transitionsanzeige **q**="gT" ("Action=Reaction" statt ⌛-Anzeige) und leitet sie mit der Überwachungszeit **upd_delay**=".." im Hintergrund an den Server weiter.

Ein Flag zur Verkürzung der Transitionsanzeige kann in <**Node**> definiert werden und dient dazu, die Update-Verzögerung im Client automatisch zu optimieren. Gültige Einträge sind:

<**DxCnfLoc**>
  <**Node nn**=".." **fast_update**="[true|false]" [..]/>

Der Standardwert ist **fast_update**="true", dies bedeutet, dass die Update-Verzögerung und damit die Transition **q**="gT" nur so lange ansteht, bis ein Wert mit einem neueren Zeitstempel als dem zuletzt empfangenen vom Server übermittelt wird. Das ist normalerweise dann der Fall, wenn der geänderte Zustand tatsächlich vom Server zurückgemeldet wird. Nur wenn der Zustand nicht - oder mit dem alten Zeitstempel zurückgemeldet wird, dann läuft die Update-Verzögerung ab und der alte (zuletzt empfangene) Zustand wird wieder angezeigt.

Die Einstellung **fast_update**="true" bedeutet auch, dass bei gleichzeitiger Bedienung mehrerer Clients, jeweils der zuletzt eingegebene Wert mit Transition **q**="gT" an allen Bedienstationen angezeigt wird, weil dieser Wert ggf. einen neueren Zeitstempel hat als der lokal eingegebene. D.h. die Update-Verzögerung wird bei Eintreffen eines Wert mit neuerem Zeitstempel aufgehoben und man sieht ggf. den aktuellen Eingabewert einer anderen Bedienstation solange in Transition **q**="gT" bis der Wert nicht mehr verändert und vom Server ohne Transitionsanzeige zurückgemeldet wird.

Mit **fast_update**="false" sieht man ggf. einen später eingegebenen Wert einer anderen Bedienstation immer erst nach Ablauf der lokalen Update-Verzögerung. D.h. mit dieser Einstellung ist während der aktiven Transition **q**="gT" nicht ersichtlich ob ggf. ein anderer Wert in den Server geschrieben wurde, der dann gilt. Generell gilt immer der zuletzt veränderte Wert im Server als "aktuell".

- *Anwendung*

Die Update-Verzögerung bzw. die Transitionsanzeige ist automatisch wirksam für bi-direktionale Datenpunkte (Read/Write) im Client <**CX**><**P nIa**=".." **r**=".." **w**=".." /></**CX**>, wenn der Datenpunkt im selben Client verändert wurde. Ohne bi-direktionale Datenpunkte benötigt man für einen Sollwert mit Anzeige zwei getrennte Datenpunkte im Client, einen zum Schreiben <**CX**><**P nIa**=".." **w**=".." /></**CX**> (Write only) und einen für die Anzeige <**CX**><**P nIa**=".." **r**=".." /></**CX**> (Read only).

☞ Die Reaktionszeit von der Eingabe bis zur Anzeige entspricht der minimalen Update-Verzögerung:
upd_delay = 2 * Netzübertragungszeit + **rd_interval** + **wr_interval** + Server Reaktionszeit
(z.B. 2 * 100 [ms] + 200 [ms] + 200 [ms] + 200 [ms] pro Verbindung)

Bei zwei gestaffelten DxNode müsste man z.B. 1.6 [sec] ohne Transitionsanzeige warten bis ein Befehl zurückbestätigt würde. Zudem würde der Eingabewert bei externer Veränderung nicht nachgeführt (gelesen) oder bei nicht erfolgreicher Übertragung "hängen bleiben" d.h. nicht auf den zuletzt gültigen Wert zurückgestellt wie für bi-direktionale Datenpunkte. Deshalb empfiehlt sich, für Anzeigen Read only, für Eingaben bi-direktionale, und nur in Ausnahmefällen Write only Datenpunkte zu definieren, wenngleich DxNode alle Varianten unterstützt.

Die Update-Verzögerung kann für jede Verbindung zur Laufzeit über den ➲ internen Datenpunkt [Prefix][ConnectName].value.**upd_delay** abgefragt werden.

Das Flag zur Verkürzung der Transitionsanzeige kann zur Laufzeit über den ➲ internen Datenpunkt [Prefix][NodeName].cmdio.**fast_update** verändert werden. Die parametrierte Vorgabe **fast_update**=".." wird dabei nicht verändert.

Die Anzahl der sich in Transition befindlichen Datenpunkte kann zur Laufzeit über den ➲ internen Datenpunkt [Prefix][NodeName].value.active_trans angezeigt werden.

v="Value" siehe auch t=".." q=".." f=".." s=".." i=".." u=".." x=".." c=".."

- *Definition*

Der eigentliche Prozess-, Mess-, Soll- oder Befehlswert stellt den wichtigsten Teil der Elementdaten dar. Alle übrigen Elementdaten beziehen sich auf diesen Wert. Gültige Darstellungen sind:

<**D**|**E**|**e** **v**="[anyValueOrText]" [**f**="..."][..]/>

Der Standardwert im ist **v**="[Nullstring]", das bedeutet "kein Wert". Das Standardformat ist **f**="%s" (string), d.h. mit Ausnahme von Hexadezimalwerten **f**="%X" oder Zeitangaben **f**="[%y|%h|%yh]" sind alle Daten ohne spezielle Formatangabe eindeutig definiert.

Für **Arrays** ist die ASCII-Zeichenfolge " | " (#x20#x7C#x20) als Delimiter reserviert, ein Array wird also z.B. mit <**D** **v**="Wert1 | Wert2 | Wert3 | WertX"/> dargestellt.

Alle Elementdaten Value, Timestamp, Quality, Format, Status, Index, Unit, Text und Event Counter beschreiben zusammen jeweils ein Ereignis bzw. einen Datenpunkt-Wert, der als **v**="..." dargestellt wird. Die aktuellsten Daten sind jeweils in DxNode gespeichert und können über Verbindungen abgerufen werden.

Elementdaten können auch als Standardwerte in der <**DPList**> oder in den Steuerelementen <**Link**...> pro Datenpunkt-Auswahl <**P**> mit <**D**|**E**|**e**> vorgegeben werden. Gültige Einträge sind:

```
<DPList>
   <Group gn="...">
      <P a|n="..."><E [v="..."][t="..."][q="..."][f="..."][s="..."][i="..."][u="..."][x="..."]/></P>
```

und/oder zum einstellen der lokalen Elementdaten bei Ausführung von ...

```
<Link1st> und/oder
<LinkOn> und/oder
<LinkOff>
   <P a|n="..." gn="..."><D|E|e [v="..."][t="..."][q="..."][f="..."][s="..."][i="..."][u="..."][x="..."]/></P>
```

Elementdaten sind grundsätzlich gleichberechtigte, optionale Eigenschaften, die in Telegrammen <**X0**><**P**> zum Teil automatisch ergänzt (z.B. **t**="..." **q**="...") und - sofern vorhanden - gemäss **attr**="..." übermittelt werden. Die Standardeinstellung ist **attr**="-a-n+v+t+q.f.s.i.u.x-c" d.h. normalerweise werden Value, Timestamp und Quality immer übertragen, die restlichen Eigenschaften nur bei Änderung, zur Initialisierung oder bei entsprechender Abfrage. Gültige Darstellungen in Telegrammen sind:

```
<X0>
   <P a|n=".."><D|E|e [v=".."][t=".."][q=".."][f=".."][s=".."][i=".."][u=".."][x=".."][c=".."]/></P>
</X0>
```

Zur Darstellung der Elementdaten sind die ASCII-Zeichen #x9 [TAB], #xA [$^{L}_{F}$], #xD [$^{C}_{R}$] und #x20..#x7F zulässig sowie alle weiteren Zeichen des jeweils netzweit gültigen Zeichensatzes z.B. **ISO-8859-1**.

Folgende ASCII-Zeichen sind für XML reserviert und daher für String-Werte **v**="..", **u**=".." oder **x**=".." durch die vorgegebenen XML-Entities (Escape-Sequenzen) zu ersetzen:

HEX- ASCII-Zeichen	**XML**	**Reservierte XML Entities**	
#x22 ["]	&quot;	Anführungszeichen (quotation mark)	Diese Zeichen sind immer durch die vorgegebenen XML-Entities (Escape-Sequenzen), dargestellt als &...; zu ersetzen
#x26 [**&**]	&amp;	Und-Zeichen (ampersand)	
#x27 [']	&apos;	Anführungsstrich (apostroph)	
#x3C [**<**]	&lt;	Kleiner-Zeichen (less than)	
#x3E [**>**]	&gt;	Grösser-Zeichen (greater than)	
HEX- ASCII-Zeichen	**XML**	**Wildcard Zeichen**	
#x2A [*]	*	Sternchen (asterisk="any number of any char")	Diese Zeichen dürfen in Textwerten frei verwendet werden
#x3F [**?**]	?	Fragezeichen (questionmark="any char")	
HEX- ASCII-Zeichen	**XML**	**Reservierte Sonderzeichen**	
#x23 [**#**]	#	Kreuzzeichen in XML Entity (sharp/number sign)	Diese Zeichen dürfen in Textwerten frei verwendet werden bzw. sind als Array Delimiter zu verwenden
#x24 [**$**]	$	Dollarzeichen für Markierung (dollar sign)	
#x3B [**;**]	;	Strichpunkt für Auflistung (semicolon)	
#x20#x7C#x20 [**|**]	|	Array Delimiter in **v**=".." (space+bar+space)	

v="Value" stellt den eigentlichen **Prozess-, Mess-, Soll- oder Befehlswert** dar. Weitere mögliche Elementdaten sind (siehe Detailbeschreibung im entsprechenden Kapitel):

t="Timestamp" stellt den **Zeitstempel** im dateTime Format dar (entspricht **f**="%yh"). Die Zeit ist Universal Time Coordinated (UTC), d.h. weltweit eindeutig, ohne Zeitzonenverschiebung oder Sommer/Winterzeitumschaltung. Der Standardwert ist **t**="1970-01-01T00:00:00". Mit Telegrammen <**X0**><**P**> übermittelte Werte ohne Zeitstempel werden bei Empfang mit der Knotenzeit gestempelt.

tc="Timestamp Corrective" zeigt eine **Zeitstempelkorrektur** in [ms] als int an. DxNode korrigiert ungültige, mit Telegrammen <**X0**><**P**> übermittelte Zeitstempel automatisch und merkt sich die korrigierte Zeit als **tc**="..". Der Standardwert ist **tc**="0".

q="Quality" stellt die **Messwertqualität** als enumerated (string) dar. Der Initialwert beim Starten von DxNode ist **q**="bWD" (bad Waiting for Initial Data). Mit Telegrammen <**X0**><**P**> übermittelte Werte ohne Qualität werden bei Empfang in DxNode automatisch auf **q**="g" (good) gesetzt.

f="Format" definiert **Format** und **Datentyp** als enumerated (string) für den Wert gemäss printf Format Specifier %x.yz in C++ (z.Z. ohne flags, width, precision). Der Standardwert ist **f**="%s" (string).

s="Status" stellt einen **User Status** als unsignedByte dar. Der Standardwert ist **s**="0". Der Status entspricht einer anwendungsseitig definierbaren, erweiterten Messwertqualität (Vendor Specific Quality).

i="Index" stellt einen **Meldungsindex** für eine indizierte Nachricht als int dar. Ein Index dient der eindeutigen Markierung bzw. Identifikation von bestimmten Meldungen. Der Standardwert ist **i**="0". Der Index ist anwendungsseitig zu definieren und anzuwenden.

u="Unit" definiert die **Masseinheit** als enumerated (string). Der Standardwert ist **u**="[Nullstring]". Eine Liste der zulässigen Masseinheiten kann im XML Schema für die Parametrierung definiert werden.

x="Text" stellt eine zusätzliche **Text Information** als string dar. Der Standardwert ist **x**="[Nullstring]". Der Text kann als beliebige Information zu den Elementdaten verstanden werden z.B. Beschreibung des Datenpunktes.

c="Counter" (reserved) wird von DxNode als **Ereigniszähler** mitgeführt. Der Zähler wird beim Start von DxNode auf Null gestellt und kann zur Laufzeit nur abgefragt werden (Read Only). Der Standardwert ist **c**="0".

- *Anwendung*

Telegramme <**X0**><**P**> mit Elementdaten <**D**|**E**|**e**> für Value, Timestamp, Quality, Format, Status, Index, Unit, Text und Ereigniszähler werden allgemein wie folgt dargestellt:

```
<X0>
   <P a|n=".."><D|E|e [v=".."][t=".."][q=".."][f=".."][s=".."][i=".."][u=".."][x=".."][c=".."]/></P>
</X0>
```

Dabei sind grundsätzlich alle Attribute der Elementdaten <**D**|**E**|**e**> optional. D.h. man kann auch nur einen neuen Zeitstempel (z.B. als Trigger) oder die veränderte Quality übermitteln. Normalerweise übermittelt eine externe Anlage jedoch mindestens den Messwerte mit **v**="..":

... entweder als spontanes Ereignis (Event, Unsolicited Read):

```
<X0><P a=".."><E v=".." /></P></X0>
```

Wir nennen das **Event Read** from Server <**E**>. In diesem Falle ist bei Verbindungseröffnung eine Synchronisation der Datenpunkte vorzunehmen. Ereignisse <**E**> werden spontan an die aktiven Subskriptionen <**SX**> im lokalen DxNode weitergeleitet und dann über die entsprechenden Verbindungen an die Clients <**CX**> übermittelt.

... oder als kontinuierliche Übermittlung mit Datenvergleich in DxNode (Polling, Triggered Read):

```
<X0><P a=".."><D v=".." /></P></X0>
```

Wir nennen das **Data Read** from Server <**D**>. In diesem Falle ist keine Synchronisation notwendig, da das Prozessabbild periodisch übermittelt wird. Daten <**D**> werden von DxNode automatisch auf Änderung geprüft und nur bei Ungleichheit als Ereignisse <**E**> an die aktiven Subskriptionen <**SX**> im lokalen DxNode weitergeleitet um dann über die entsprechenden Verbindungen an die Clients <**CX**> übermittelt zu werden.

Die Anbindung kann von DxNode auch Sollwerte **v**=".." empfangen (Advise, Exception Write):

```
<X0><P a="."><e v=".." t=".." q=".." /></P></X0>
```

Wir nennen das **Event Write** to Server <**e**>. In diesem Falle ist keine Synchronisation notwendig, da das Prozessabbild ja von der Anbindung gehalten wird. Für Sollwerte und Befehle werden immer Value, Timestamp und Quality als Ereignis <**e**> übermittelt, sofern die Elementdaten nicht mit **attr**=".." beschränkt werden.

verbose="Trace Flags" (DxMonitor)

- *Definition und Anwendung*

DxNode erlaubt das Beobachten und Aufzeichnen von Transaktionen incl. Fehler-, Warnungs- und Informationsmeldungen über eine normale Verbindung <**Connect**>. Dazu ist im Telegramm <**X0**> bei der Verbindungsherstellung die Meldungstiefe mit Attribut **verbose**=".." für die Ausgabe zu definieren. Gültige Einträge sind:

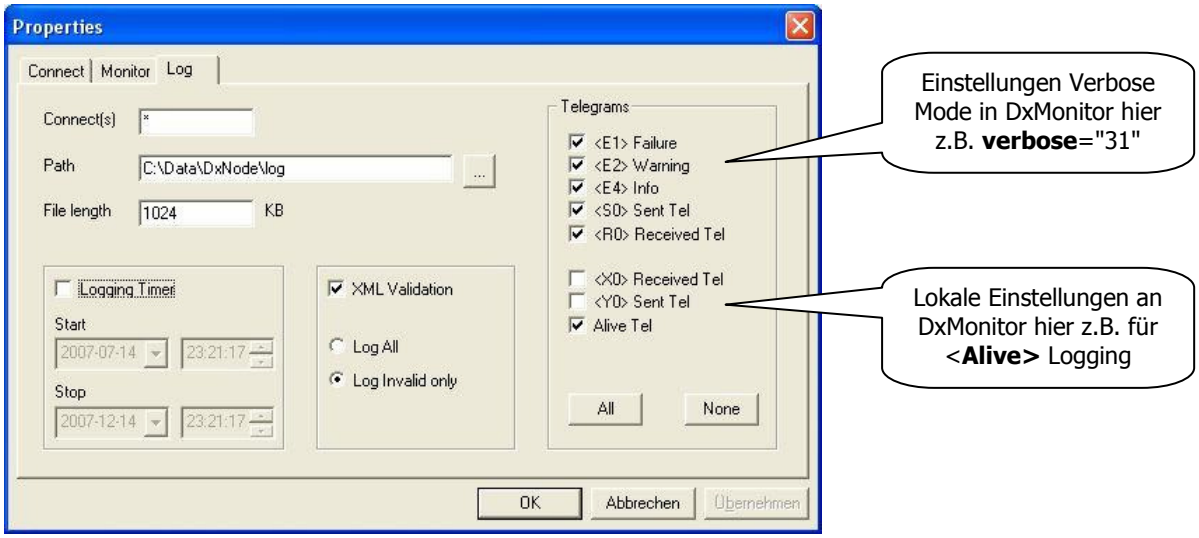
```
<X0><Connect cn=".." [verbose="[0..31]"][..]/>[<CX>|<SX>]</X0>
```

Der Standardwert ist **verbose**="0", dies bedeutet, dass keine Beobachtungsmeldungen von DxNode an den DxMonitor bzw. den Teilnehmer gesendet werden. Folgende Werte sind einstellbar:

Beobachtungsmeldung **von DxNode ...**	**verbose**="⇳"	**Aufzeichnung mit DxMonitor**
<**E1 t**=".." [..] **msg**="[anyText]" />	"1"	DxNode meldet <**E1**> Fehler (Error Level 1)
<**E2 t**=".." [..] **msg**="[anyText]" />	"2"	DxNode meldet <**E2**> Warnung (Error Level 2)
<**E4 t**=".." [..] **msg**="[anyText]" />	"4"	DxNode meldet <**E4**> Information (Error Level 4)
<**S0 t**=".." [..]><**X0**> **t**=".." [..]</**X0**></**S0**>	"8"	DxNode sendet <**X0**> an Partner (Send)
<**R0 t**=".." [..]><**X0**> **t**=".." [..]</**X0**></**R0**>	"16"	DxNode empfängt <**X0**> von Partner (Receive)
Meldungen von/an DxMonitor **		**Aufzeichnung mit DxMonitor**
<**Y0 t**=".." [..]></**Y0**>	n/a	DxMonitor schreibt Befehl <**X0**> an DxNode (Write)
<**X0 t**=".." [..]></**X0**>	n/a	DxMonitor liest Daten <**X0**> von DxNode (Read)
<**X0 ...**><**Alive**><**AliveR**></**X0**>	n/a	**Alive**-Meldungen in allen Telegrammen <**X0**>

** Lokale Einstellungen an DxMonitor, diese werden nicht mit verbose=".." an DxNode übermittelt.

Die Werte können auch kombiniert d.h. als Summe eingegeben werden z.B. schaltet der Wert "7" die Meldungen <**E1**>, <**E2**>, <**E4**> ein oder "24" die Meldungen <**S0**>, <**R0**> oder "31" alle Flags. Das Attribut **msg**=".." enthält die Klartextmeldung. Die echten Telegramme <**X0**> vom bzw. an DxNode werden mit Element <**X0**> 1:1 wiedergegeben. Die Meldungen <**Y0**> und <**X0**> stellen Ein/Ausgaben an DxMonitor dar, sie werden von DxNode mit entsprechenden Telegrammen <**R0**> bzw. <**S0**> bestätigt. Die Aufzeichnung erfolgt z.B. in **Datei C:\Data\DxNode\log\DxLog_[host]_[datetime].xml:**



Fehlermeldungen <**E1**> (Failure) bedeuten, dass die angeforderte Funktion von DxNode nicht erfüllt werden konnte, z.B. Konfigurationsdatei nicht gefunden oder bei Zugriff auf korrupte interne Daten.

Warnungsmeldungen <**E2**> (Warning) zeigen wichtige Ereignisse wie Trace Messages in den Steuerelementen <**Link1st**>, <**LinkOn**>, <**LinkOff**> z.B. <**E2 t**=".." **msg**="*** Link Initialization ***"/> oder sie deuten auf mögliche Fehler hin die jedoch keine unmittelbaren Folgen für DxNode haben z.B. ungültiges Telegramm eines Teilnehmers.

Informationsmeldungen <**E4**> (Information) zeigen zulässige Operationen an, die keine direkten Folgen für DxNode haben z.B. neuer Datenpunkt <**P**> anlegen.

Zusammen mit einer Subskription <**CX**>, <**SX**> können nahezu beliebige Abfragen für DxMonitor definiert werden.

version="XML Version"

- *Definition und Anwendung*

Die Konfigurationsdateien sollen mit einer so genannten Processing Instruction <?**xml**?> die Version des XML Standards und den verwendeten Zeichensatz (Encoding) mitführen. Gültige Einträge sind:

<?**xml version**="1.0" **encoding**=".."?>
<**DxCnfLoc**> oder <**DxCnfExt**>

Die Standardwerte für die XML Processing Instruction sind **version**="1.0" und **encoding**="UTF-8", dies bedeutet, dass die Datei dem XML Standard Version 1.0 entspricht und die Daten mit dem Zeichensatz Unicode Transformation Format 8-Bit dargestellt sind. Für Westeuropa ist der ISO Latin-1 Zeichensatz d.h. **encoding**="ISO-8859-1" sinnvoll weil die meisten Anbindungen das Unicode Transformation Format nicht unterstützen und damit alle String-Variablen umwandeln müssten.

w="Write to Server" siehe r=".."

- *Definition und Anwendung*

Read from Server **r**=".." bzw. Write to Server **w**=".." definieren die Datenpunktbezeichnung im Server für eine Subskription <**CX**> und/oder <**SX**> oder für eine Abfrage <**QX**>. Die Eingaben definieren, ob ein Datenpunkt über die entsprechende Verbindung nur gelesen (Read only), nur geschrieben (Write only) oder gelesen UND geschrieben (Read/Write) werden soll. Gleichzeitig ermöglichen die Einträge eine Umbenennung (Renaming) der Datenpunkte zwischen Client und Server. Für gültige Eingaben ➲ siehe **r**="Read from Server".

wr_interval="Write Interval" siehe rd_interval=".."

- *Definition und Anwendung*

Leseintervall **rd_interval**=".." und Schreibintervall **wr_interval**=".." definieren eine minimale Wartezeit in [ms] zwischen zwei aufeinander folgenden Telegrammen <**X0**><**P**> mit Elementdaten <**D**|**E**> (Read from Server) bzw. <**e**> (Write to Server) für Verbindungen mit einer Subskription <**CX**> oder <**SX**> (Dauerauftrag). Die Parameter wirken nur auf den echten Datenaustausch, wobei jedes Telegramm eine nicht eingeschränkte Anzahl von Datenpunkten und Elementdaten <**P** <**D**|**E**|**e** ... /></**P**> enthalten kann, andere Telegramme z.B. zur Subskription unterliegen nicht einem Intervall d.h. sie werden spontan gesendet. Für gültige Eingaben ➲ siehe **rd_interval**="Read interval".

x="Text Information" siehe auch v=".."

- *Definition und Anwendung*

Die Text Information bezieht sich auf den Wert **v**=".." und kann bei Bedarf in der <**DPList**> oder in den Steuerelementen <**Link**...> pro Datenpunkt-Auswahl <**P**> als <**D**|**E**|**e** **x**=".." /> vorgegeben werden. Gültige Einträge sind:

<**D**|**E**|**e** [**v**=".."] **x**="[anyText]" [..]/>

Der Standardwert ist **x**="[Nullstring]". Der Text kann als beliebige Information zu den Elementdaten verstanden werden z.B. Beschreibung des Datenpunktes.

Zur Darstellung der Elementdaten sind die ASCII-Zeichen #x9 [TAB], #xA [$^{L}_{F}$], #xD [$^{C}_{R}$] und #x20..#x7F zulässig sowie alle weiteren Zeichen des jeweils netzweit gültigen Zeichensatzes z.B. **ISO-8859-1**.

Folgende ASCII-Zeichen sind für XML reserviert und daher für Texte **x**=".." durch die vorgegebenen XML-Entities (Escape-Sequenzen) zu ersetzen:

HEX- ASCII-Zeichen	**XML**	**Reservierte XML Entities**	
#x22 ["]	&quot;	Anführungszeichen (quotation mark)	Diese Zeichen sind immer durch die vorgegebenen XML-Entities (Escape-Sequenzen), dargestellt als &...; zu ersetzen
#x26 [**&**]	&amp;	Und-Zeichen (ampersand)	
#x27 [']	&apos;	Anführungsstrich (apostroph)	
#x3C [**<**]	&lt;	Kleiner-Zeichen (less than)	
#x3E [**>**]	&gt;	Grösser-Zeichen (greater than)	
HEX- ASCII-Zeichen	**XML**	**Wildcard Zeichen**	
#x2A [*****]	*	Sternchen (asterisk="any number of any char")	Diese Zeichen dürfen in Textwerten frei verwendet werden
#x3F [**?**]	?	Fragezeichen (questionmark="any char")	
HEX- ASCII-Zeichen	**XML**	**Reservierte Sonderzeichen**	
#x23 [**#**]	#	Kreuzzeichen in XML Entity (sharp/number sign)	Diese Zeichen dürfen in Textwerten frei verwendet werden
#x24 [**$**]	$	Dollarzeichen für Markierung (dollar sign)	
#x3B [**;**]	;	Strichpunkt für Auflistung (semicolon)	
#x20#x7C#x20 [**|**]	|	Array Delimiter in **v**=".." (space+bar+space)	

Anhang

OPC Unified Architecture und DxNode.Net

Der **OPC Standard** ➲ http://www.opcfoundation.org besteht aus Teil-Spezifikationen als insgesamt über 1000-seitiges Werk. Der Standard basiert ursprünglich auf der Microsoft™ DCOM-Technologie, die mit der Spezifikation OPC Unified Architecture (OPC UA) sukzessive durch XML und Web-Services abgelöst wird.

OPC Unified Architecture (UA) ab 2007	OPC Spezifikationen seit 1998 basierend auf DCOM
OPC UA Part 1 Concepts	
OPC UA Part 2 Security Model	OPC Security
OPC UA Part 3 Address Space Model	OPC Common, OPC Core Components etc.
OPC UA Part 4 Services	
OPC UA Part 5 Information Model	
OPC UA Part 6 Mappings	
OPC UA Part 7 Profiles	
OPC UA Part 8 Data Access	OPC Data Access (DA), OPC XML-DA, OPC Data eXchange (DX)
OPC UA Part 9 Alarms	OPC Alarms & Events (AE)
OPC UA Part 10 Commands	OPC Commands
OPC UA Part 11 Historical Access	OPC Historical Data Access (HDA)
…	OPC Batch
…	OPC Complex Data

DxNode.Net ist keine Spezifikation, sondern eine komplette, lauffähige Software die in den beteiligten Systemen installiert wird und ohne Programmierung parametriert werden kann. DxNode wurde von Anfang an auf TCP/IP, XML und Web-Services ausgelegt und unterstützt eine zu OPC kompatible Datendarstellung mit Value **v**="..", Timestamp **t**=".." und Quality **q**=".." (VTQ). Die Zeitstempel sind nach ISO-8601 in [ms] formatiert, die Qualitätsangaben enthalten alle von OPC und sind um einige erweitert.

DxNode.Net verfügt über OPC Client und Server Schnittstellen und unterstützt alle wesentlichen Merkmale, die mit der OPC UA Spezifikation (Unified Architecture) geplant sind.

Die wichtigsten Unterschiede zu OPC

Topic	OPC Spezifikation	DxNode.Net
Connect	Nur Client kann verbinden	Client oder Server kann Verbindung herstellen
Data Access	Client Subskription und Synchron Access (Last Event Polling) ohne Wildcards	Client und/oder Server Subskription (Events) und Single Request (Last Event Polling) mit Wildcards
Alarms&Events	Client Subskription (Events)	Standard Transport wie Data Access
Security	Zusatzspezifikation	Parametrierbar pro Connect und Zugangsport
Store&Forward	Nicht spezifiziert	Parametrierbar pro Connect und Datenpunkt
Redundanz	Nicht spezifiziert	Automatische Umschaltung für Multiple Servers
Chaining	Nicht spezifiziert	Beliebige Client/Server⇔Client/Server Ketten
Bi-direktionaler Datenaustausch	Gängige Methode, senden an Device und warten auf Rückmeldung	Gängige Methoden plus Action=Reaction mit Transitionsanzeige
Read/Write	Single Channels	Aufteilbar oder Single Channels
Renaming	Nicht spezifiziert	Datenpunkt Rangierung wird unterstützt
Datenfilter	Nicht spezifiziert	Zeit und Status Filter werden unterstützt
Addressing/Naming	Nicht spezifiziert	Zwei parametrierbare gleichwertige Adressräume
Data Transport	(D)COM oder XML mit OPC UA	TCP/IP und XML
Parametrierung	Nicht spezifiziert	Spezifizierte XML-Datei
Monitoring	Nicht spezifiziert	DxMonitor für alle Transaktionen

DxNode unterstützt ausser den herkömmlichen Methoden für Datenzugriffe mit Subskription und Einzelabfrage eine zusätzliche bi-direktionale Übertragungsart mit Transitionsanzeige. Dabei spiegelt der lokale DxNode eine Eingabe im Client sofort als "Action=Reaction" (statt ⌛-Anzeige), während der Befehl im Hintergrund an den Server weitergeleitet - und zurückgemeldet wird.

Die Subskriptions- und Transportmethoden sind bei DxNode so gewählt, dass Ereignisse übermittelt werden können. DxNode ist daher in der Lage so genannte Alarms&Events zu transportieren, die bei OPC getrennt verarbeitet werden.

DxNode unterstützt zwei gleichwertige parametrierbare Adressräume. Die Datenpunkte können in Client und Server unterschiedlich bezeichnet und direkt mit der Subskription umbenannt werden (Renaming). Datenpunkte können parametrierbar mit Wildcards ausgewählt werden. Ein Datenpunkt im Client kann zwei getrennte Datenpunkte für Read und Write im Server haben. Man kann damit das Netzwerk auf einen Namen pro Datenpunkt standardisieren und für die Anbindung zwei Adressen (rd+wr) definieren.

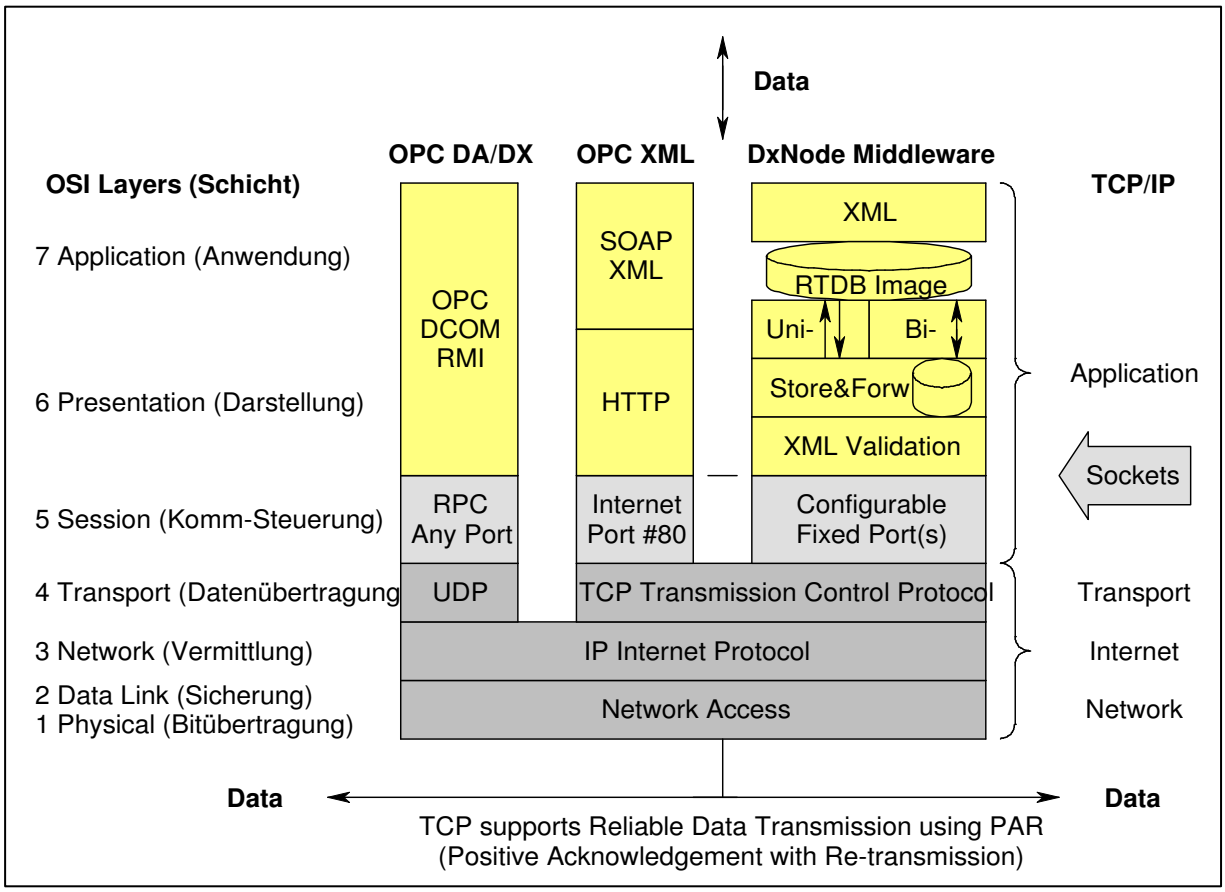
DxNode kann Meldungen Speichern und Weiterleiten (Store&Fwd), parametrierbar pro Verbindung und Datenpunkt. DxNode unterstützt eine parametrierbare automatische Umschaltung auf mehrere andere Server bei Verbindungsunterbruch (Multiple Server Redundancy). Knoten können in beliebigen Ketten hintereinander geschaltet und verknüpft werden (Chaining)

Der Datentransport zwischen DxNode sowie zwischen DxNode und Anbindungen erfolgt über TCP/IP und XML. Dabei können alle Transaktionen mit DxMonitor beobachtet werden.

Verbindungen können vom Server oder vom Client hergestellt werden, bei OPC kann das nur der Client.

Die gesamte Parametrierung incl. diejenige für die externe Anbindung ist als XML-Datei definierbar.

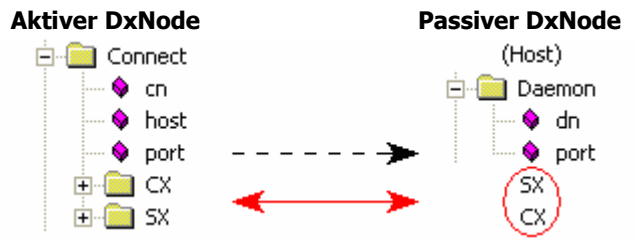
Vergleich der Datenübertragung im OSI Modell



DxNode Client/Server Verbindungen

Der aktive DxNode stellt jeweils die Verbindung zum passiven DxNode her, im passiven DxNode muss dazu ein Zugangsport <**Daemon**> definiert sein, der aktive DxNode benötigt dazu kein Port. Die Subskription für Client <**CX**> oder Server <**SX**> kann im aktiven oder passiven DxNode parametriert werden. DxNode unterstützt drei Varianten für die Konfiguration von Verbindungen <**Connect**>:

1. XML Struktur - Einseitig parametrierte Verbindung mit <CX/SX> im aktiven DxNode

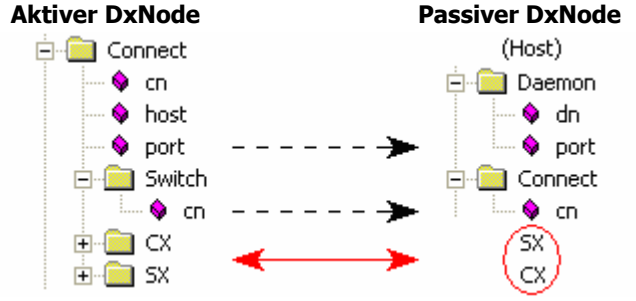


Typische Anwendung: DxMonitor oder wenn die Verbindung im passiven DxNode nicht überwacht werden muss.

Diese Methode entspricht der gängigen Subskription für einen Client <**CX**> an einen passiven Server. DxNode unterstützt jedoch mit derselben Methode auch die Subskription für einen Server <**SX**> an einen passiven Client.

Der passive DxNode (Host) benötigt nur den Zugangsport <**Daemon**>, die Verbindung bleibt hier weitgehend unüberwacht, nur die Anzahl der Verbindungen ist im passiven DxNode bekannt.

2. XML Struktur - Beidseitig parametrierte Verbindung mit <CX/SX> im aktiven DxNode

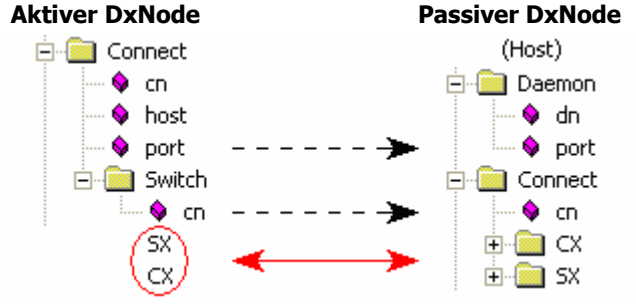


Typische Anwendung wenn die Verbindung zusätzlich im passiven DxNode überwacht/gesteuert werden soll.

In diesem Fall definiert der aktive Knoten, welche Datenpunkte mit der Subskription <**CX**> und <**SX**> ausgetauscht werden. Die Methode ist wie 1.) oben, sie erlaubt jedoch die beidseitige Steuerung, Sperrung und Überwachung der Verbindung mit automatisch erzeugten Datenpunkten **a**="[cn]..", indem mit Befehl <**Switch**> auf den im passiven Knoten vorgesehenen <**Connect**> umgeschaltet wird.

Für Store&Forw Daten mit Redundanzumschaltung ist Methode 3.) unten zu wählen. Um die Konfiguration zu vereinheitlichen, empfiehlt sich dann, nur Methode 3.) als Standard zu verwenden.

3. XML Struktur - Beidseitig parametrierte Verbindung mit <CX/SX> im passiven DxNode



Typische Anwendung für Store&Forw Daten mit Überwachung der Verbindung im passiven DxNode.

In diesem Fall definiert der passive Knoten, welche Datenpunkte mit der Subskription <**CX**> und <**SX**> ausgetauscht werden. Die Methode ist grundsätzlich wie 2.) oben, erlaubt jedoch die Konfiguration für eine Redundanzumschaltung mit Store&Forw Daten aus dem passiven DxNode.

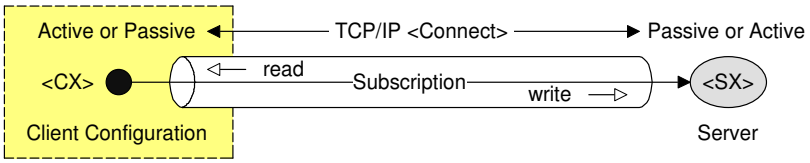
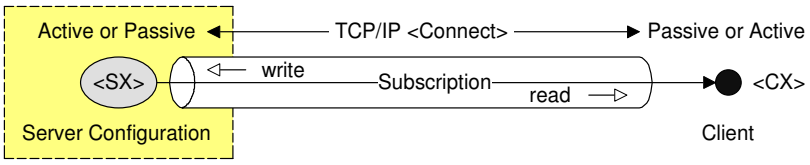
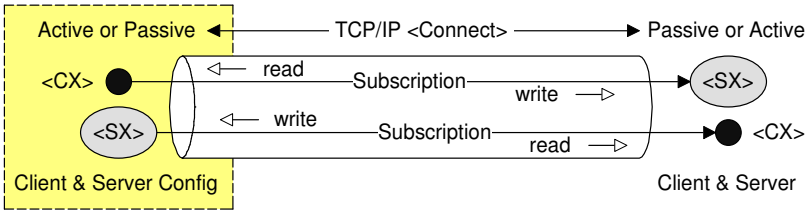
Die Konfiguration ist auch typisch für eine externe Anbindung am passiven Knoten. Das externe System spielt dann die Rolle des aktiven DxNode und schaltet mit dem Befehl <**Switch**> auf den <**Connect**> mit <**CX**> und/oder <**SX**>im passiven Knoten. Siehe auch Beispiel ➲ Verbindungsaufbau und –ablauf.

Legende:

☞ Client <**CX**> und/oder Server <**SX**> werden jeweils nur für einen Partner DxNode (aktiv oder passiv) definiert, der andere Partner übernimmt automatisch die komplementäre Server und/oder Client Rolle.

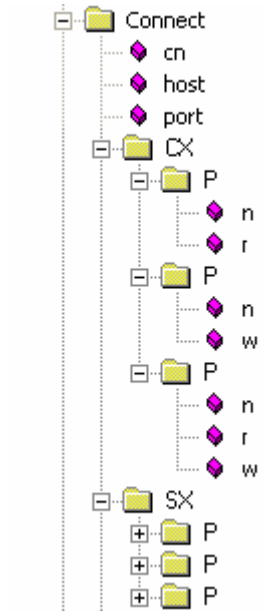
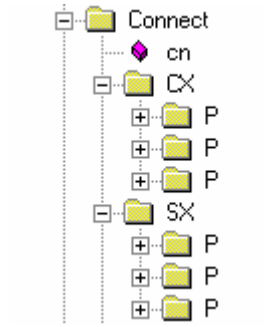
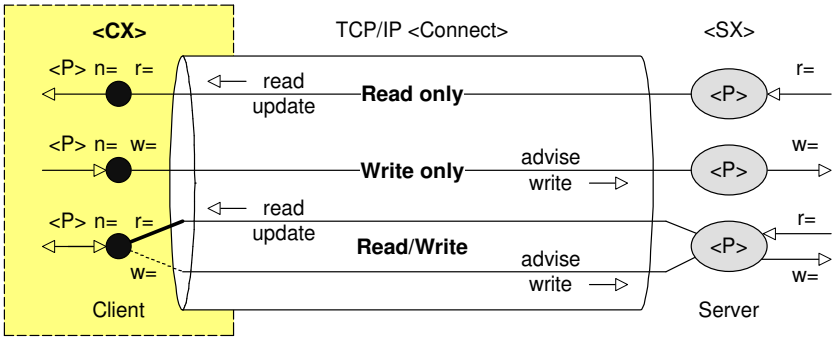
☞ Der aktive DxNode benötigt kein Zugangsport <**Daemon**>, er stellt die Verbindung zum passiven DxNode her. Ein aktiver DxNode ohne Port ist nicht zugänglich für fremde Eindringlinge.

Konfiguration für Client <CX> und/oder Server <SX> Subskription (alle Varianten)

XML Konfiguration	<Connect> Client <CX> im lokalen DxNode
Verbindung aktiv mit CX: Connect cn host port CX ... oder passiv mit CX: Connect cn CX attr P P P	 Element <**CX**> definiert eine Client Subskription für eine Auswahl von Datenpunkten die im lokalen DxNode genutzt werden aber dem Partner als Server (owner) gehören. Nach dem Verbindungsaufbau übernimmt der Partner automatisch die komplementäre Server Rolle <**SX**>. Mit **attr**=".." werden die zu kommunizierenden Attribute festgelegt. Wichtig! mit der Initialisierung werden die zu lesenden (read) Datenpunkte des lokalen DxNode <**CX**> auf die Werte des Partners <**SX**> synchronisiert. Elemente <**P**> definieren die zu übertragenden Datenpunkte die, je nach Konfiguration, gelesen und/oder geschrieben werden können (➲ siehe Read/Write weiter unten). Das Element <**CX**> kann als Alternative auch im Element <**Matrix**> definiert werden.
XML Konfiguration	**<Connect> Server <SX> im lokalen DxNode**
Verbindung aktiv mit SX: Connect cn host port SX ... oder passiv mit SX: Connect cn SX attr P P P	 Element <**SX**> definiert eine Server Subskription für eine Auswahl von Datenpunkten die dem lokalen DxNode gehören (owner) und vom Partner als Client genutzt werden. Nach dem Verbindungsaufbau übernimmt der Partner automatisch die komplementäre Client Rolle <**CX**>. Mit **attr**=".." werden die zu kommunizierenden Attribute festgelegt. Wichtig! mit der Initialisierung werden die zu lesenden (read) Datenpunkte des Partners <**CX**> auf die Werte des lokalen DxNode <**SX**> synchronisiert. Elemente <**P**> definieren die zu übertragenden Datenpunkte die, je nach Konfiguration, gelesen und/oder geschrieben werden können (➲ siehe Read/Write weiter unten). Das Element <**SX**> kann als Alternative auch im Element <**Matrix**> definiert werden.
XML Konfiguration	**<Connect> Client <CX> und Server <SX> im lokalen DxNode**
Verbindung aktiv CX + SX: Connect cn host port CX SX ... oder passiv CX + SX: Connect cn CX P P P SX P P P	 Pro Verbindung können eine Client <**CX**> und eine Server <**SX**> Subskription definiert werden. Nach dem Verbindungsaufbau übernimmt der Partner automatisch die komplementären Server und Client Rollen. Wichtig! mit der Initialisierung werden die zu lesenden (read) Datenpunkte der entsprechenden <**CX**> auf die Werte von <**SX**> synchronisiert. Elemente <**P**> definieren die zu übertragenden Datenpunkte die, je nach Konfiguration, gelesen und/oder geschrieben werden können (➲ siehe Read/Write weiter unten). Die Elemente <**CX**> und <**SX**> können als Alternative auch im Element <**Matrix**> definiert werden.

☞ Der Verbindungsaufbau erfolgt unabhängig von der Kommunikationsart <**CX**/**SX**> oder **r**=".."/**w**=".." und wird einzig mit den Attributen **host**=".." und **port**=".." für den aktiven <**Connect**> bestimmt.

Konfiguration für "Read from Server" und/oder "Write to Server"

XML Konfiguration	<CX> und <SX> mit Read r=".." und/oder Write w=".."
Verbindung aktiv CX + SX:  ... oder passiv CX + SX:  	Die nachstehende Beschreibung gilt für Client <**CX**> und Server <**SX**> gleichermassen. Die Elemente <**P**> definieren die zu kommunizierenden Datenpunkte. Sie können explizit angegeben - oder mittels Attributen der Datenpunktliste <**DPList**> **gn**="..", **n**=".." oder **a**=".." gefiltert werden. Dabei dürfen die Ausdrücke ".." auch Wildcards (*?) enthalten z.B. <**P n**="Test*" /> bedeutet "alle Datenpunkte mit Namen Test...". Als Beispiel sei hier eine Client Subskription <**CX**> dargestellt:  Die Attribute **r**=".." und **w**=".." bedeuten, unabhängig von <**CX**> oder <**SX**>, immer "**Read from Server**" (Server lesen) und "**Write to Server**" (Server schreiben). Die dargestellten Datenpunkte zeigen die drei möglichen Konfigurationen bezüglich Read und Write: 1. der erste Datenpunkt hat **Read only** Zugriff. Der Wert wird bei Änderung im Server zum Client übermittelt und bei der Initialisierung automatisch auf den aktuellen Wert des Servers synchronisiert. 2. der zweite Datenpunkt hat **Write only** Zugriff. Der Wert wird bei gezielter Änderung (advise) im Client zum Server übermittelt jedoch NIE automatisch auf den aktuellen Wert des Clients synchronisiert. 3. der dritte Datenpunkt hat **Read**/**Write** Zugriff (bi-direktional). Dies bedeutet, dass derselbe Datenpunkt sowohl gelesen wie auch geschrieben werden kann. Es handelt sich um eine Kombination von Read und Write wobei der Wert des Servers grundsätzlich dominiert aber vom Client gezielt verändert werden kann (advise). Dabei wird für die Zeit der Übertragung im Client ein Statusanzeige "in Transition" erzeugt und ggf. automatisch wieder der alte Wert eingestellt - sollte der geänderte Wert im Server nicht empfangen und akzeptiert worden sein. Die Methode erlaubt damit gleichzeitig zwei Festlegungen: a) das **Zugriffsrecht im Server** (Access Right), nämlich ob ein Datenpunkt gelesen und/oder geschrieben werden darf, d.h. Zugriffsrecht Read only, Write only oder Read/Write hat und ... b) die **unterschiedliche Bezeichnung in Client und Server** (Umbenennung). Für Read/Write Signale (bi-direktional) können bei Bedarf im Server getrennte Read und Write Datenpunkte definiert werden z.B. für Befehl und Befehls-Rückmeldung. Das Gleichheitszeichen "=" als Attributwert bedeutet "gleicher Name" in Client und Server d.h. keine Umbenennung. Jeder Datenpunkt kann also mit <**CX**>, <**SX**> und <**P**> für jede Verbindung individuell konfiguriert werden. Weil Datenpunktbezeichnungen auch Wildcards (*?) zulassen, bedeutet z.B. der XML Ausdruck ... <**Connect cn**="XY"><**CX**><**P n**="*" **r**="=" **w**="=" /></**CX**></**Connect**> etwa folgendes: **Verbinde** "XY" als **Client** für alle "*" **Datenpunkte** mit **Read**/**Write** Zugriff und gleicher "=" Bezeichnung im Server. Der XML Ausdruck stellt nicht nur die Konfiguration dar, sondern er ist gleichzeitig auch das Telegramm zur Subskription der Datenpunkte beim Partner, wobei nur <**CX**> in <**SX**> gewandelt werden muss !

☞ Die Attribute **r**=".." und **w**=".." bedeuten, unabhängig von <**CX**> oder <**SX**>, immer "**Read from Server**" (vom Server lesen) bzw. "**Write to Server**" (zum Server schreiben).

DxNode Interne Datenpunkte

DxNode erzeugt automatisch die in den folgenden Tabellen aufgeführten Datenpunkte

Bezeichnungskonvention

[Name].cmdio.sss Befehls-Ein/Ausgabe (Bi-direktional Read/Write)
[Name].matrix.sss Matrix-Information pro Verbindung (Bi-direktional bzw. Read Only)
[Name].perf.sss Performance-Werte (Read Only) werden zyklisch mit <**Node perf_cycle**=".." /> ausgegeben
[Name].value.sss Wert-Ausgabe (Read Only)
| |
| Erweiterung = **Attribut-Name**, nachstehend **fett für parametrierbare Items**
Name aus <**Node nn**="[NodeName]">, <**Daemon dn**="[DaemonName]"> oder <**Connect cn**="[ConnectName]"> kann linksbündig mit Präfix aus <**Node prefix**="[Prefix]"> (Standardwert = Null-String) erweitert werden.

Datenpunkte Allgemein <Node>

Lokale Adresse a=".."	**Type**	**rd/wr**	**Verweis auf Attribut, Beschreibung**
[Prefix][NodeName].cmdio.code	%s	rd/wr	Lizenzschlüssel für DxNode-Lizenz
[Prefix][NodeName].cmdio.serial_no	%s	rd/wr	Beliebige Serie-Nummer für Anwender (optional), wird mit Lizenzschlüssel verknüpft
[Prefix][NodeName].cmdio.state	%L	rd/wr	Status von DxNode: **2** = **running as service** **1** = **running as application** **0** = **restart application** **-99** = **shutdown** / **terminate** steuert alle Elemente <**Connect**><**Link1st**>
[Prefix][NodeName].cmdio.**fast_update**	%b	rd/wr	➲ siehe auch **fast_update**=".." Transitionsanzeige: true→fast, false→upd_delay Der Standardwert ist **fast_update**="true"
[Prefix][NodeName].cmdio.**perf_cycle**	%i	rd/wr	➲ siehe auch **perf_cycle**=".." Sollwert Aktualisierungsrate der Statistik in [ms] Der Standardwert ist **perf_cycle**="10000" [ms]
[Prefix][NodeName].perf.bytes_rcv	%i	rd	Statistik / Daten empfangen in [Byte/sec]
[Prefix][NodeName].perf.bytes_snd	%i	rd	Statistik / Daten gesendet in [Byte/sec]
[Prefix][NodeName].perf.bytes_tot	%i	rd	Total Daten empfangen+gesendet in [Byte/sec]
[Prefix][NodeName].perf.interval	%i	rd	Istwert Aktualisierungsrate der Statistik in [ms]
[Prefix][NodeName].perf.tel_rcv	%i	rd	Statistik / Anzahl Telegramme/sec empfangen
[Prefix][NodeName].perf.tel_snd	%i	rd	Statistik / Anzahl Telegramme/sec gesendet
[Prefix][NodeName].perf.tel_tot	%i	rd	Total Telegramme/sec empfangen+gesendet
[Prefix][NodeName].value.active_addr	%i	rd	Anzahl aktive Datenpunkte mit Adressen **a**=".."
[Prefix][NodeName].value.active_connects	%i	rd	Anzahl aktive Verbindungen <**Connect**>
[Prefix][NodeName].value.active_items	%i	rd	Anzahl aktive Datenpunkte mit **a**=".." oder **n**=".."
[Prefix][NodeName].value.active_names	%i	rd	Anzahl aktive Datenpunkte mit Namen **n**=".."
[Prefix][NodeName].value.active_trans	%i	rd	Anzahl Datenpunkte in Transition mit **q**="gT"
[Prefix][NodeName].value.arguments	%s	rd	Argumentenliste bei Start von DxNode
[Prefix][NodeName].value.computer_name	%s	rd	Rechner Name
[Prefix][NodeName].value.dp_version	%s	rd	➲ siehe auch **config_version**=".." Konfigurationsversion der <**DPList**>
[Prefix][NodeName].value.hw_code	%s	rd	Hardware Code für DxNode-Lizenz
[Prefix][NodeName].value.license_type	%L	rd	Lizenz 0 = Demo, 1 = Lizenziert/ok, 2 = Zeit-Limit
[Prefix][NodeName].value.start_time	%yh	rd	Datum/Zeit Start von DxNode
[Prefix][NodeName].value.vendor	%s	rd	Hersteller-Informationen DxNode
[Prefix][NodeName].value.version	%s	rd	Versionsbezeichnung des DxNode Standardprogramms DxNode.exe (Executable)

Performance-Werte "...perf..." werden periodisch mit [Prefix][NodeName].cmdio.perf_cycle ausgegeben

Datenpunkte pro Zugangsport <Daemon>

Lokale Adresse a=".."	**Type**	**rd/wr**	**Verweis auf Attribut, Beschreibung**
[Prefix][DaemonName].cmdio.state	%L	rd/wr	Zugangsport Status: **x** = **x partner connected** **etc.** **2** = **2 partner connected** **1** = **1 partner connected** **0** = **restart / initializing** **-1** = **stop / disable** **-99** = **shutdown / terminate** steuert ALLE über diesen Zugangsport erstellte Verbindungen <**Connect**> d.h. wenn dieser Port z.B. auf stop/disabled gestellt wird, dann werden die entsprechenden Verbindungen ebenfalls stop/disabled geschaltet
[Prefix][DaemonName].value.connects	%s	rd	Anzahl der aktiven bzw. inaktiven Verbindungen als Bit-Muster dargestellt z.B. 1110001000 bei vier aktiven Connects und **max_conn**="10"
[Prefix][DaemonName].value.**max_conn**	%i	rd	➲ siehe auch **max_conn**=".." Max. Anzahl zugelassene Verbindungen Der Standardwert ist **max_conn**="10" (definiert String-Länge von ".value.connects")
[Prefix][DaemonName].value.message	%s	rd	Klartext Statusinformation des Listener Ports
[Prefix][DaemonName].value.**port**	%s	rd	➲ siehe auch **port**=".." Lokales Listener Port Adresse oder Service Nr. Der Standardwert ist **port**="7581/tcp"
[Prefix][DaemonName].value.start_time	%yh	rd	Datum/Zeit Neustart des Listener Ports

Datenpunkte pro Anschluss/Verbindung <Connect>

Lokale Adresse a=".."	**Type**	**rd/wr**	**Verweis auf Attribut, Beschreibung**
[Prefix][ConnectName].cmdio.state	%L	rd/wr	Verbindungsstatus: **x** = **x partner connected** **etc.** **2** = **2 partner connected** **1** = **1 partner connected** **0** = **restart / initializing** **-1** = **stop / disable** **-99** = **shutdown / terminate** steuert die ➲ Elemente <**LinkOff**><**LinkOn**>
[Prefix][ConnectName].cmdio.active_host	%L	rd/wr	Host-Steuerung/Anzeige bei mehreren **host**=".." **x** = **select host # x** **etc.** **2** = **select host # 2** **1** = **select host # 1** **0** = **restart auto select at host # 1** ☞ nur gültig für den aktiven Partner
[Prefix][ConnectName].cmdio.**alive**	%i	rd/wr	➲ siehe auch **alive**=".." Timeout Verbindungsüberwachung in [sec] Der Standardwert ist **alive**="30" [sec]
[Prefix][ConnectName].cmdio.**host**	%s	rd/wr	➲ siehe auch **host**=".." Host-Name oder IP-Adresse von Partner
[Prefix][ConnectName].cmdio.**port**	%s	rd/wr	➲ siehe auch **port**=".." Zugangs Port Adresse oder Service Nr. von Partner Der Standardwert ist **port**="7581/tcp"

Lokale Adresse a=".."	**Type**	**rd/wr**	**Verweis auf Attribut, Beschreibung**
[Prefix][ConnectName].cmdio.**reconnect_cycle**	%i	rd/wr	➲ siehe auch **reconnect_cycle**=".." Zeitintervall zum Wiederherstellen der Verbindung nach Unterbruch bzw. Umschaltung in [sec]. Der Standardwert ist **reconnect_cycle**="1" [sec]
[Prefix][ConnectName].matrix.cx	%s	rd/wr	Client-Matrix <**CX**> als Teil-String ohne "<CX" dargestellt (nur für Testzwecke)
[Prefix][ConnectName].matrix.**cx_version**	%s	rd	➲ siehe auch **config_version**=".." Konfigurationsversion Client-Matrix <**CX**>
[Prefix][ConnectName].matrix.sx	%s	rd/wr	Server-Matrix <**SX**> als Teil-String ohne "<SX" dargestellt (nur für Testzwecke)
[Prefix][ConnectName].matrix.**sx_version**	%s	rd	➲ siehe auch **config_version**=".." Konfigurationsversion Server-Matrix <**SX**>
[Prefix][ConnectName].perf.bytes_rcv	%i	rd	Statistik / Daten empfangen in [Byte/sec]
[Prefix][ConnectName].perf.bytes_snd	%i	rd	Statistik / Daten gesendet in [Byte/sec]
[Prefix][ConnectName].perf.bytes_tot	%i	rd	Total Daten empfangen+gesendet in [Byte/sec]
[Prefix][ConnectName].perf.interval	%i	rd	Istwert Aktualisierungsrate der Statistik in [ms]
[Prefix][ConnectName].perf.tel_rcv	%i	rd	Statistik / Anzahl Telegramme/sec empfangen
[Prefix][ConnectName].perf.tel_snd	%i	rd	Statistik / Anzahl Telegramme/sec gesendet
[Prefix][ConnectName].perf.tel_tot	%i	rd	Total Telegramme/sec empfangen+gesendet
[Prefix][ConnectName].value.host_ip	%s	rd	➲ siehe auch **host**=".." Nur IP-Adresse von Partner
[Prefix][ConnectName].value.last_rcv	%yh	rd	➲ siehe auch <**X0 t**=".."> und <**X0 tgt**=".."> Zeitstempel des zuletzt empfangenen Telegrammes
[Prefix][ConnectName].value.last_snd	%yh	rd	➲ siehe auch <**X0 t**=".."> Zeitstempel des zuletzt gesendeten Telegrammes
[Prefix][ConnectName].value.message	%s	rd	Klartext Statusinformation der Verbindung
[Prefix][ConnectName].value.**rd_interval**	%i	rd	➲ siehe auch **rd_interval**=".." Leseintervall in [ms]→ Read from Server Der Standardwert ist **rd_interval**="200" [ms]
[Prefix][ConnectName].value.start_time	%yh	rd	Datum/Zeit Start der Verbindung
[Prefix][ConnectName].value.**upd_delay**	%i	rd	➲ siehe auch **upd_delay**=".." und **q**="gT" Client Update-Verzögerung (Transition) in [ms]. Der Standardwert ist **upd_delay**="3000" [ms]
[Prefix][ConnectName].value.**wr_interval**	%i	rd	➲ siehe auch **wr_interval**=".." Schreibintervall in [ms]→ Write to Server Der Standardwert ist **wr_interval**="200" [ms]

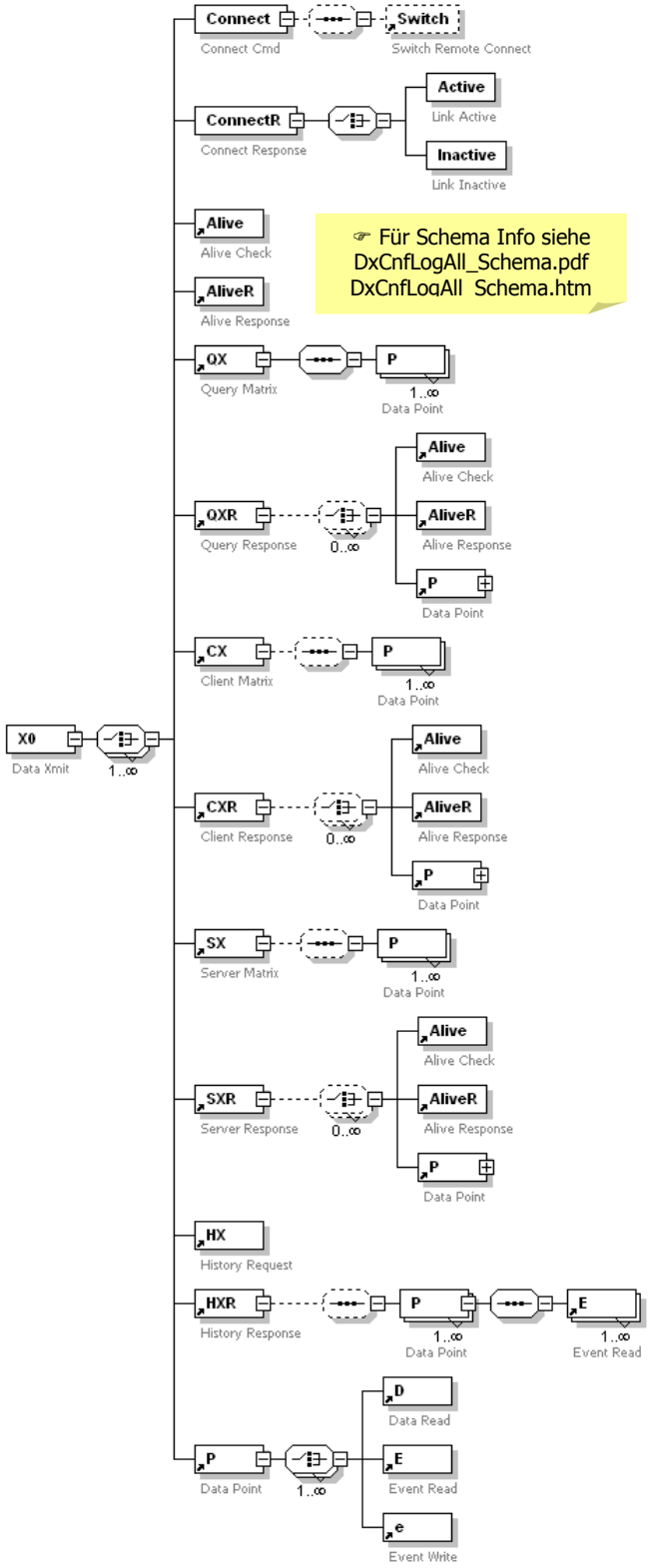
Performance-Werte "...perf..." werden periodisch mit [Prefix][NodeName].cmdio.perf_cycle ausgegeben

Knotenstatus und Verbindungszustände

Value [..][..].cmdio.state	[..][**Node**].cmdio.state	[..][**Daemon**].cmdio.state	[..][**Connect**].cmdio.state	
x = indication only	n/a	**x partner connected**	**x partner connected**	**Link On**
etc.	etc.	etc.	etc.	
2 = indication only	**running/service**	**2 partner connected**	**2 partner connected**	
1 = indication only	**running/application**	**1 partner connected**	**1 partner connected**	
0 = command/indication	**restart application**	**restart/initializing**	**restart/initializing**	**Link Off**
-1 = command/indication	n/a	**stop/disable**	**stop/disable**	
-99 = command/indication	**shutdown/terminate**	**shutdown/terminate**	**shutdown/terminate**	
Bemerkungen	steuert alle Elemente <**Link1st**>	steuert zugeordnete Elemente <**Connect**>	steuert die Elemente <**LinkOff**><**LinkOn**>	
	☞ Der Shutdown-Befehl stoppt den entsprechenden Prozess unwiderruflich d.h. er kann nicht wieder mit [Prefix][...].cmdio.state gestartet werden			

DxNode Telegrammstrukturen und XML Schema

Alle zulässigen Telegramme sind mit der folgenden Grafik schematisch dargestellt. Die dazu verwendeten XML Elemente sind weitgehend die gleichen wie für die Konfiguration. Die detaillierte Darstellung ist mit dem XLM Schema **DxCnfLogAll.xsd** definiert. Das Schema (siehe DxCnfLogAll.pdf) beschreibt auch die gesamte DxNode Konfiguration sowie die DxMonitor Log Dateien.



☞ Für Schema Info siehe DxCnfLogAll_Schema.pdf DxCnfLogAll_Schema.htm

- Verbindungsherstellung
  <**Connect**> Request
  <**ConnectR**> Response

- Verbindungsüberwachung
  <**Alive**> Heartbeat Signal
  <**AliveR**> Response
  (Alive-Meldung kann auch in Response-Telegrammen eingebettet werden)

- Single Query / Request
  <**QX**> Query Matrix
  <**QXR**> Query Response mit
  <**P**> wie Datenaustausch unten

- Client Subscription
  <**CX**> Client Matrix
  <**CXR**> Client Response mit
  <**P**> wie Datenaustausch unten

- Server Subscription
  <**SX**> Server Matrix
  <**SXR**> Server Response mit
  <**P**> wie Datenaustausch unten

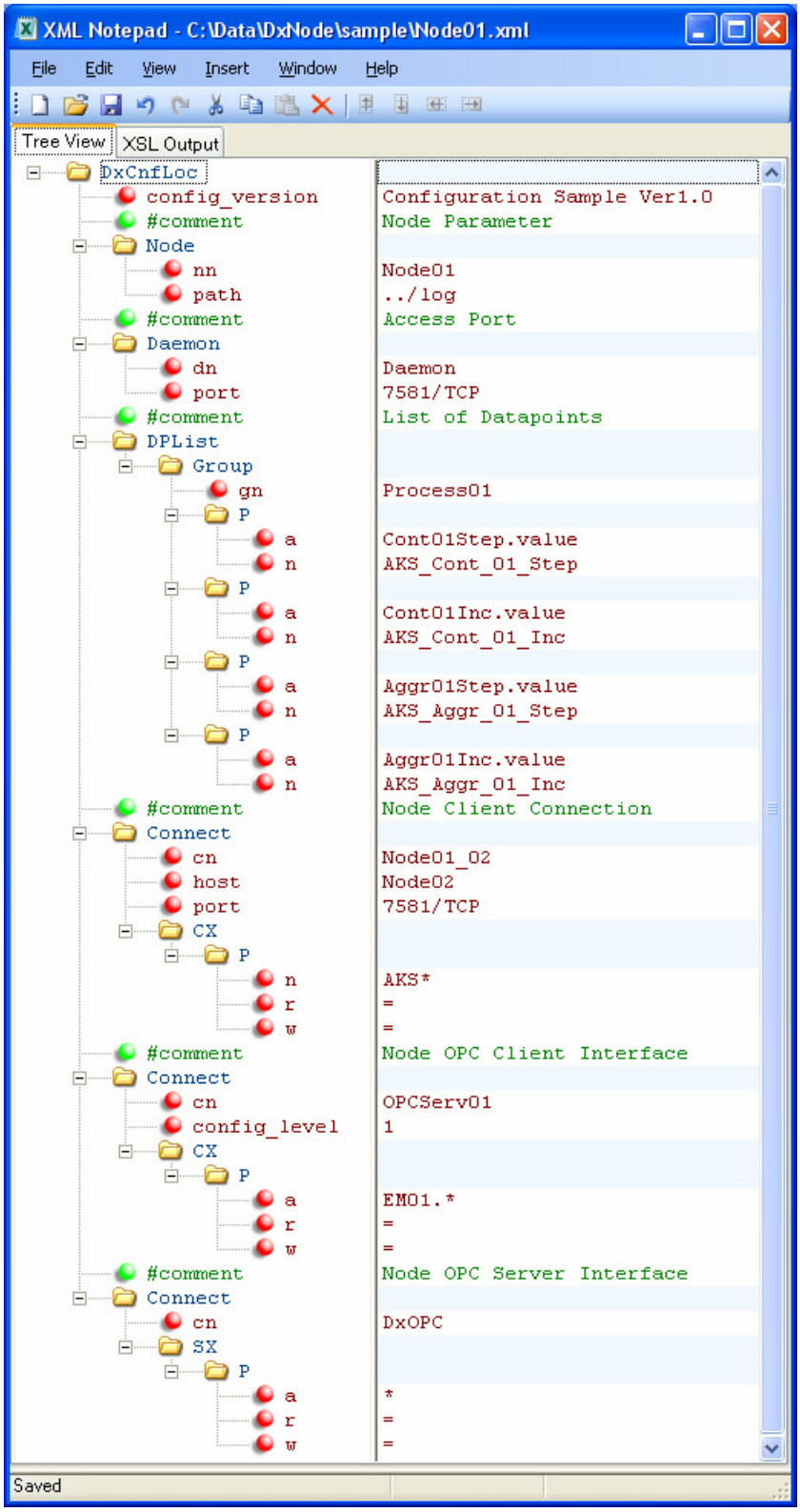
- Server Store&Forward
  <**HX**> History Request
  <**HXR**> History Response
  <**P**> nur mit <**E**>

- Datenaustausch
  <**P**> Datapoint
  <**D**> Data Polling/Synchronisation
  <**E**> Event Read from Server
  <**e**> Event Write to Server

☞ Für einfache Anbindungen genügen die grau markierten Funktionen

Konfigurationsbeispiele

Beispiel einer kompletten, einfachen Konfiguration unter Verwendung von Microsoft XML Notepad:



<**Node nn**="Node01"> Name

<**Daemon port**="7581/TCP">

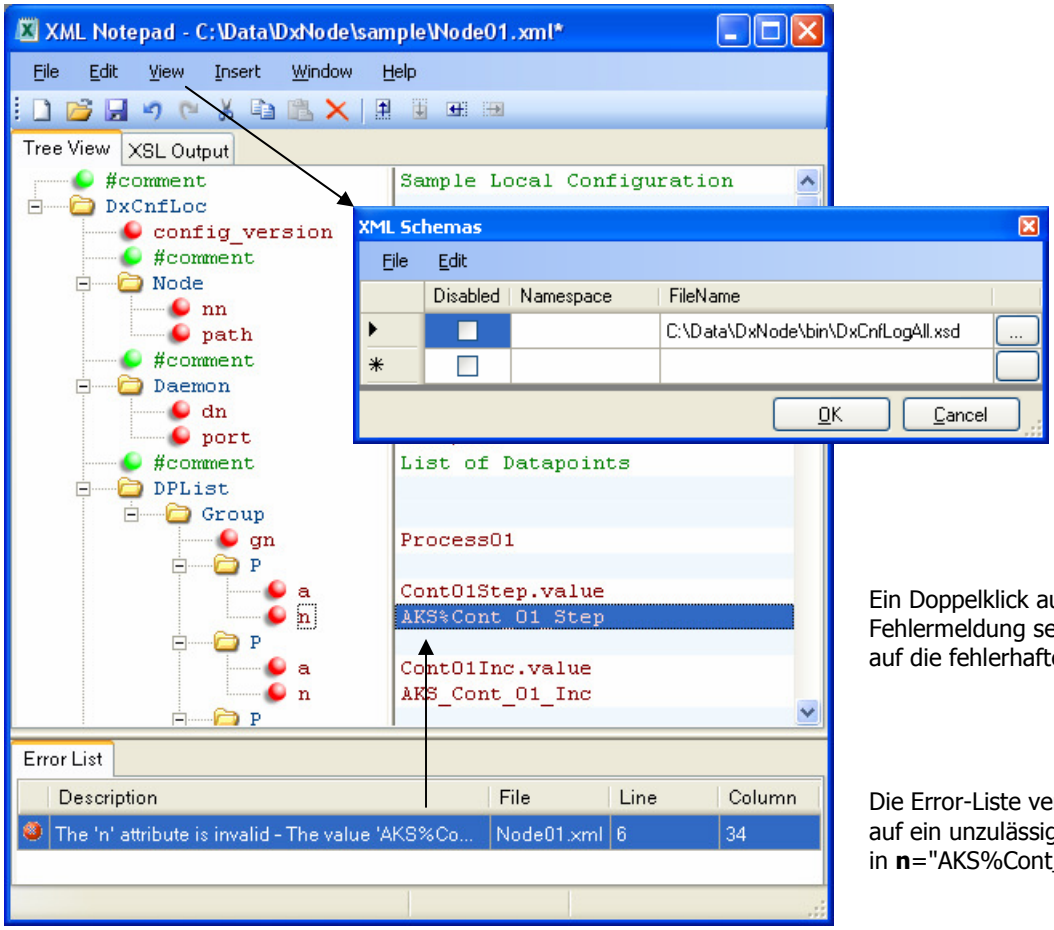
<**DPList**> Datenpunktliste hat <**Group gn**="Process01"> mit 4 Datenpunkten <**P**> mit den Adressen **a**=".." und den Namen **n**="..."

<**Connect cn**="Node01_02"> verbindet aktiv **host**="Node02" und **port**="7581/TCP".
DxNode spielt hier die Rolle des Clients <**CX**> gegenüber dem Partner

<**Connect cn**="OPCServer01"> ist passiv.
DxNode spielt hier die Rolle des Clients <**CX**> am externen OPC Server

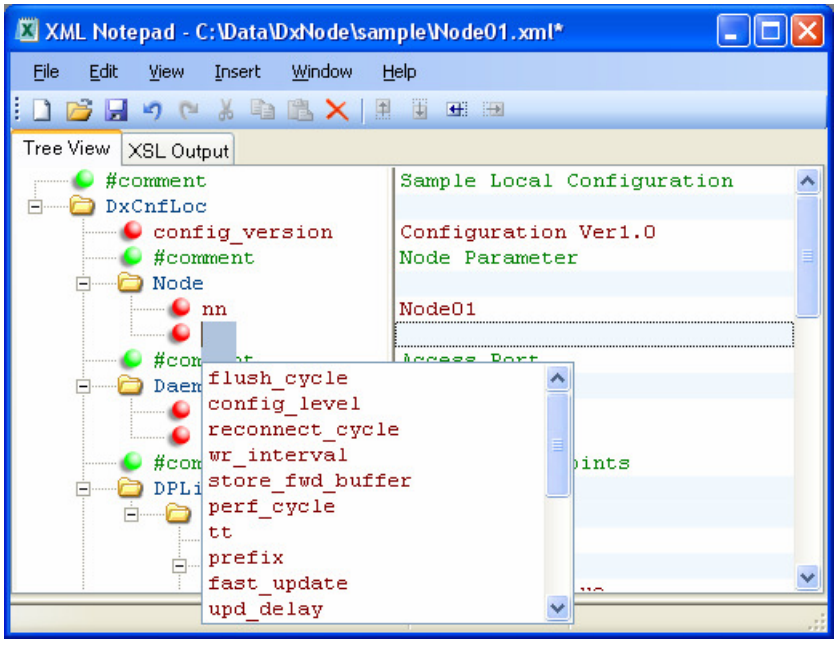
<**Connect cn**="DxOPC "> ist passiv.
DxNode spielt hier die Rolle des Servers <**SX**> für den DxOPC Server.

Mit **View→Schemas**... kann man das XML Schema **DxCnfLogAll.xsd** anfügen, um die Konfiguration oder auch Telegramm Log-Dateien zu validieren:



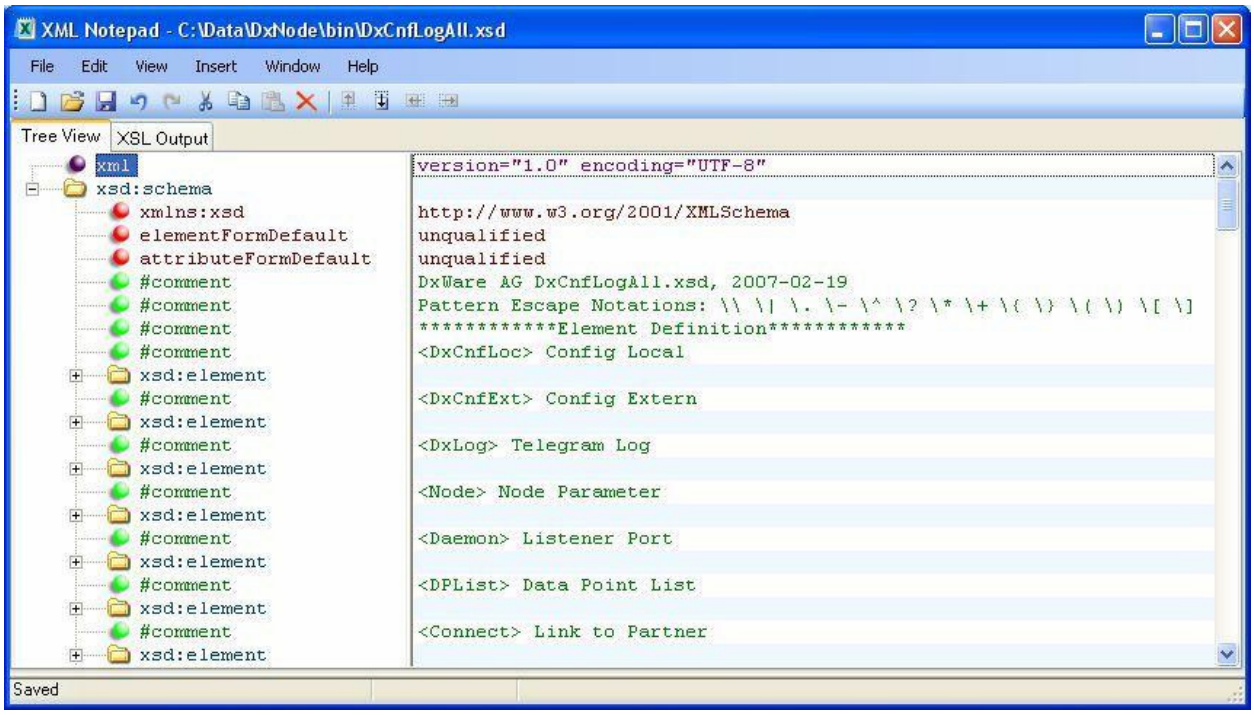
Ein Doppelklick auf die Fehlermeldung setzt der Cursor auf die fehlerhafte Zeile.

Die Error-Liste verweist hier z.B. auf ein unzulässiges Zeichen "%" in **n**="AKS%Cont_01_Step".

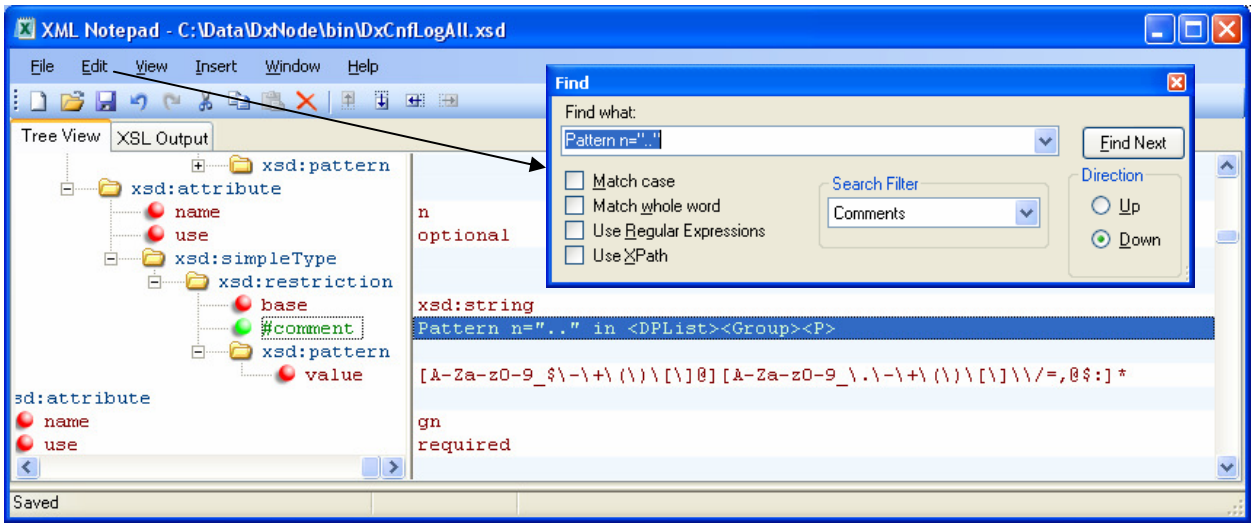


Das Schema hilft auch bei der Eingabe mit Auswahllisten, hier z.B. die Liste mit den zulässigen Parametern für <**Node**>.

Auch das XML Schema **DxCnfLogAll.xsd** kann mit dem Microsoft XML Notepad dargestellt werden:



Das XML Schema enthält einige Mustervorlagen (Pattern) um zulässige Zeichen in Adressen und Namen zu beschränken und z.B. keine ÄÖÜ zuzulassen oder der IEC-1131 zu genügen. Zum Anpassen eines Patterns sucht man die richtige Position im Schema z.B. für den DP Namen mit **Edit→Find** **n**=".." in der Kommentarzeile, dann korrigiert man den nachfolgenden **value**="[...]" mit dem gewünschten Ausdruck:



Zum Beispiel sei der Ausdruck oben "[A-Za-z0-9_$\-\+\(\)\[\]@][A-Za-z0-9_\.\-\+\(\)\[\]\\/=,@$:]*" um die Zeichen ÄÖÜ am Anfang und um ÄÖÜäöü für die weiteren Zeichen zu ergänzen, dann wäre der Ausdruck neu wie folgt: "[A-ZÄÖÜa-z0-9_$\-\+\(\)\[\]@][A-ZÄÖÜa-zäöü0-9_\.\-\+\(\)\[\]\\/=,@$:]*"

☞ Wichtig! Es gilt der XML Standard gemäss http://www.w3.org/XML.
Die Pattern Escape Notations \\ \| \. \- \^ \? * \+ \{ \} \(\) \[\] sind zu berücksichtigen. Alle anderen Zeichen werden entsprechend dem eingestellten **encoding**=".." gewandelt, hier z.B. UTF-8.
Es müssen alle Stellen im Schema korrigiert werden, z.B. kommt **value**="[...]" für **n**=".." mehrmals vor.

DxNode Installation als Applikation oder Service

Die Installation von DxNode erfolgt unter Windows normalerweise im Verzeichnis C:\Programme\DxNode. DxNode benötigt ausser dem Standardprogramm **DxNode.exe** (Executable) eine in XML dargestellte Konfigurationsdatei (Parametrierung), die mindestens die zum Starten relevanten Informationen enthält. Auf diese, so genannte lokale Konfigurationsdatei ist beim Start zu verweisen. Der folgende Systemaufruf startet DxNode z.B. mit der Konfigurationsdatei **Node01.xml**. Mit dem Argument **-app** wird DxNode als **User Applikation** mit einem **Systemfenster** für Meldungsausgaben gestartet:

C:\Programme\DxNode\bin**DxNode.exe** X:\Konfiguration**Node01.xml -app**

Mit Argument -install wird DxNode als **Service** ohne Systemfenster installiert. DxNode startet dann jeweils beim Hochfahren des Computers mit der gleichen Konfigurationsdatei, hier z.B. mit **Node02.xml**:

C:\Programme\DxNode\bin**DxNode.exe** X:\Konfiguration**Node02.xml -install**

Meldungsausgaben sind in diesem Falle aus der Log-Datei [NodeName].Log zu entnehmen. Für Konfigurationsänderungen ist DxNode mit dem Restart Befehl "0" neu zu starten (siehe ➲ DxNode Interner Datenpunkt [Prefix][NodeName].cmdio.state). Zur De-Installation des Services muss der laufende DxNode zuerst mit dem Shutdown Befehl "-99" gestoppt werden, dann de-installieren mit:

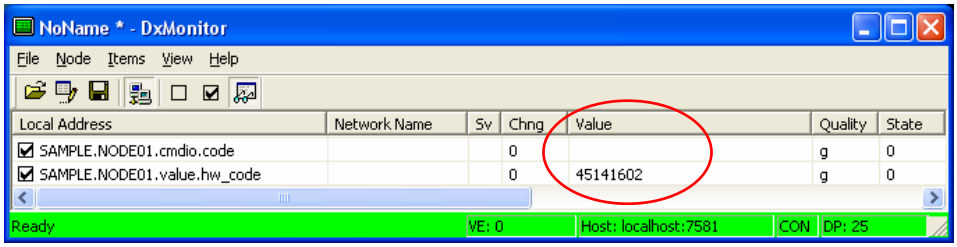
C:\Programme\DxNode\bin**DxNode.exe -uninstall**

Programm-, Konfigurations- und Arbeitsverzeichnis (siehe auch ➲ **path**="..") sind frei wählbar. Wenn kein Konfigurationsverzeichnis festgelegt wird, erwartet DxNode die Konfigurationsdatei im Arbeitsverzeichnis d.h. in dem Verzeichnis aus dem DxNode mit dem Systemaufruf gestartet wird.

DxNode Lizenzierung

Zur Lizenzierung von DxNode auf einem PC ist ein **Lizenzschlüssel** erforderlich. Dieser kann unter Angabe des **Knotennamens** und des **Hardware Codes** vom DxNode Lieferanten angefordert werden.

Der Knotenname ist mit dem Eintrag <**Node nn**="[NodeName]"/> der Konfigurationsdatei vorgegeben, der Hardware Code ist mit dem DxMonitor auf Datenpunkt [Prefix][NodeName].value.hw_code ablesbar. Der Lizenzschlüssel ist mit dem DxMonitor auf Datenpunkt [Prefix][NodeName].cmdio.code einzugeben:



Lizenzschlüssel ← Eingabe
Hardware Code → Anzeige

Optional kann der Anwender eine eigene **Serie-Nummer** als beliebige Zeichenfolge auf Datenpunkt [Prefix][NodeName].cmdio.serial_no eingeben. Die Serie-Nummer dient z.B. dazu, Systeme mit gleichen Knotennamen zu unterscheiden und ist zur Lizenzierung zusammen mit dem Knotennamen und dem Hardware Code anzugeben. Die Serie-Nummer muss gegebenenfalls vor dem Lizenzschlüssel mit dem DxMonitor eingegeben werden.

☞ **Knotenname** (nodeName), **Hardware Code** (hwCode), **Lizenzschlüssel** (licenseCode) und **Serie-Nummer** (serialNo) können auch aus der Log-Datei [NodeName].Log entnommen werden oder sind im Systemfenster ersichtlich, sofern DxNode zuvor als Applikation gestartet wurde:

```
nodeName    : NODE01 (used for licensing!)
hwCode      : 45141602 (used for licensing!)
serialNo    :
licenseCode : (invalid code)
```

Internetadressen für XML-Editoren und Tools

Adressen für XML-Editoren und Tools für die Erstellung und Bearbeitung von XML-Dateien:

Microsoft™ XML Nodepad 2007 Freeware, unterstützt Schema	http://msdn2.microsoft.com/en-us/library/aa905339.aspx oder http://www.snapfiles.com/get/xmlnotepad.html
XML Spy, geeignet für Schema	http://www.xmlspy.com
XML Pad, **XML Mind**, Freeware	http://www.wmhelp.com, http://www.xmlmind.com
XML Tools, diverse Freeware	http://www.garshol.priv.no/download/xmltools

DxNode Einbindungsworkshop

Ein Programmierer braucht - neben der klaren Vorstellung von der anlagenseitigen Anbindung - im Wesentlichen zu wissen, wie man eine TCP/IP Socket Verbindung zu einem Zugangsport herstellt und wie man den XML-Datenstrom übermittelt bzw. empfängt. Dabei hilft ihm ein einfaches Source-Beispiel, das die relevante Prozedur für eine Anbindung offen darlegt. Diese Offenheit erlaubt, eine Anbindung in jeder Programmiersprache, sofern diese TCP/IP unterstützt, z.B. VBA, Delphi etc. Selbst eine SPS-Ankopplung ist mit TCP/IP und XML realisierbar. Der Programmierer kann die Anbindung anhand des Beispiels für sein System frei gestalten, er muss sich nur an den beschriebenen Ablauf und die XML-Darstellung halten.

Für Programmierer wird empfohlen, einen eintägigen Workshop zu besuchen. Das Ziel ist eine einfache Musteranbindung mit DP lesen und DP schreiben. Die Vorbereitung umfasst, dass sich der Programmierer mit der anlagenseitigen Anbindung auskennt und einen ausgerüsteten Laptop mit TCP/IP Schnittstelle und seine Entwicklungsumgebung (z.B. Visual C++) mitbringt.

☞ Download Adresse für **DxNode**, **DxMonitor** und **Source Beispielen** für DxNode Anbindungen
http://www.DxWare.com → Downloads wählen dann → Installation DxNode

