Erstellt von Werner Krings, Tani GmbH
Erstelldatum: 1.7.2019, zuletzt überarbeitet: 14.7.2021

# Technische Daten Opc Server

## Allgemein

Der Tani OPC Server ist ein Multiprotokoll und Multiopc OPC Server. Er erlaubt den Zugriff auf Steuerungen und Geräte unterschiedlicher Hersteller. Er ist einfach zu konfigurieren. Er bietet viele Diagnosen.

## OPC Schnittstellen

- **OPC UA** (Unified Architecture)
- **OPC Pipe** Offene Schnittstelle
- **OPC DA** (Classic OPC über DCOM, nur unter Windows)

Die maximale Anzahl von OPC Clients hängt nur vom Resourcenbedarf ab. Ein PC Baujahr 2017 kann mehrere hundert Verbindungen bearbeiten.
Alle OPC Schnittstellen arbeiten lokal innerhalb eines PC oder über Netzwerk
OPC UA unterstützt das schnelle Binärprotokoll. Security wird in allen Varianten unterstützt. Multicast Discovery wird unterstützt.
Es werden DataAccess DataItems genutzt die jedes bis zu 200K lang sein können.
Bei Classic OPC wird DCOM über Netz ausdrücklich nicht empfohlen, es wird aber unterstützt.

### OPC UA functionality and limitations

The OPC UA implementation conforms to the specification 1.05.
The OPC UA Standard Model is supported, some extensions exist.
The maximum single request and answerlen is 16m
The OPC UA Alarms & Conditions module is supported. This includes filters, history.
An internal discovery server is active on standard, it supports multicast discovery also. It can be used as a global discovery server. Alternatively an external discovery server can be configured.
The certificate management GDS Push is supported.

The session timeout will be limited to one hour.
The server and client certificate will be renewed if the Tani self signed certificate is used. All other certificates remain unaffected on expiring. The certificate validity is checked all 12h. It will be renewed seven days before it expires. Running connections will not be affected, new connections will use the new certificate.
AddNodes is supported with the following restrictions:

- Reference type must be OpcUaId_Organizes
- NodeId can't be specified
- BrowseName can't contain a dot
- NodeClass must be Variable or Object
- NodeAttributes for Variable:
  - DisplayName: unspecified or equal to BrowseName
  - Description: unspecified or any text
  - Value: is ignored; new variables will always be initialized to 0 (if numeric) or "" (if string type)
  - DataType:
    - OpcUaType_Boolean
    - OpcUaType_SByte, OpcUaType_Byte
    - OpcUaType_Int16/32/64, OpcUaType_UInt16/32/64
    - OpcUaType_Float, OpcUaType_Double
    - OpcUaType_String
    - OpcUaType_LocalizedString. This will be handled outside OPC UA as a normal string. The LocaldId always is a null string
    - OpcUaType_DateTime
    - OpcUaType_ExtendedObject, OpcUaType_ExtendedObjectEx. Mostly this are structures. One of the structure types under Types -> DataTypes -> BaseDataType -> Structure -> UserStructures; these are the structures known to the PLC Engine core.
      - if the structure is given both here and via TypeDefinition, both settings must match
    - if unspecified, OpcUaType_Byte or the structure type of the TypeDefinition is used
  - ValueRank, ArrayDimensions: unspecified (= scalar), scalar or a one-dimensional array of any size
  - AccessLevel, UserAccessLevel: unspecified or (OpcUa_AccessLevels_CurrentRead | OpcUa_AccessLevels_CurrentWrite)
  - MinimumSamplingInterval: unspecified or 0
  - WriteMask, UserWriteMask: unspecified or OpcUa_NodeAttributesMask_Value
- NodeAttributes for Object:
  - DisplayName: unspecified or equal to BrowseName
  - Description: unspecified or any text
  - EventNotifier, WriteMask, UserWriteMask: unspecified or 0
- TypeDefinition for Variable:
  - OpcUaId_BaseDataVariableType
  - one of the structure types under Types -> VariableTypes -> BaseVariableType -> BaseDataVariableType -> UserStructures; these are the structures known to the PLC Engine core.
- TypeDefinition for Object:
  - OpcUaId_FolderType
- Each RPC as a calling queue of 10. If the requests are coming faster before handled they will return a memory error.

Machine models from the OPC Foundation or the VDMA directly can be loaded with its corresponding XML file.

The security certificate key minimum length are

- Basic128Rsa15: RSA Key Length 1024 .. 4096
- Basic256: RSA Key Length 1024 .. 4096
- Basic256Sha256: RSA Key Length 2048 .. 4096

Datenverkehr zwischen den OPC Schnittstellen (tunneling) ist möglich. Damit wird die OPC DA Tunnel realisiert.

## SPS Schnittstellen

Alle Steuerungen werden über Netzwerk angeschlossen. Oft ist das Ethernet, WLAN oder andere Netze. Alle Seriell Ethernet und MPI Ethernet Wandler für Steuerungen werden unterstützt.

## Konfigurationsschnittstellen

Die Konfiguration wird über die mitgelieferte Konfigurationssoftware oder über OPC mit dem System Topic vorgenommen.
Die Konfigurationsverbindung wird mit TLS 1.2 verschlüsselt. Diese Verschlüsselung kann für die Nutzung in Ländern mit Verschlüsselungsverbot abgeschaltet werden.

## Netzwerkredundanz bei Verbindungen zu Steuerungen und Geräten

Verbindungen zu Geräten und Steuerungen unterstützen Netzwerk Redundanz.
Möglich sind Doppel und Dreierredundanz.
Die Redundanzarten Dynamisch und Statisch sind möglich.
Bei **Dynamischer Redundanz** ist eine beliebige Unterverbindung der Master. Bei Unterbrechung wird eine andere Verbindung Master.
Bei **Statischer Redundanz** ist die erste Verbindung Master. Fällt die aus so wird auf eine Slave Verbindung geschaltet. Sobald die Master Verbindung wieder arbeitet wird auf diese zurückgeschaltet.

Die Verbindungen der Redundanz sollen auf unterschiedlichen Netzwerkkarten und in unterschiedlichen IP Subnetzen betrieben werden.

## SPS Typen und SPS Protokolle

- Siemens **S7 1200 und 1500** Familie. Die **optimierten Bausteine** werden untersützt, **Strukturen Alarme Ereignisse**, ebenso Browsing aller Symbole und Kommentare online. Die 2021 SPS Firmware 2.9 wird ebenfalls unterstützt, ebenso Firmware 3.x.
- Siemens **S7 200, 300 und 400, Logo 8, ET200**über RFC1006 oder Sinec H1. Siemens CPs oder der Ethernetanschluß an der CPU kann genutzt werden. Gängige MPI Gateways wie Hilscher Netlink, Helmholz Netlink, IBH Softec Netlink, INAT Echolink, Process Informatik S7Lan oder Softing Netlink werden unterstützt. Zur S7 kompatible Systeme wie VIPA Speed7 sind einsetzbar.
- Siemens S5 über RFC1006, SPS Header, RAW oder Sinec H1. Unterstützt werden Siemens CPs, INAT CPs, Helmholz CPs, IBH Softec S5Net, Process Informatik S5Lan.
- Rockwell **CompactLogix**, **ControlLogix** und **GuardLogix**, alle Ausgabestände. Rockwell **Micro 8** Serie wie die 800, 810, 820 und Weitere.
  Rockwell **PLC5** und **SLC**, alle Ausgabestände.
  **Routingpfade** werden voll unterstützt. Sie erreichen so Steuerungen z.B. über eine Control Logix und über unterlagerte Busse wie DH+. So erreichen Sie auch z.B. eine PLC5 über eine andere Rockwell SPS wie eine Compact Logix als Router.
- GE PACSystems von General Electric. Diese Steuerungsfamilie nutzt CIP von Rockwell.
- Mitsubishi **Melsec Q**, **QL** und **FX5** Familie über das SLMP Protokoll (3E Protokoll).
- **BACnet Geräte**. BACnet ist in der Gebäudetechnik weit verbreitet. BBMD und COV wird unterstützt, ebenso Alarme, Events, Trend, Calendar, Shedule, rpc (Remote Procedure Calls), Listen und Weitere.
- **KNX Geräte**. KNX wird auch im Gebäudeumfeld eingesetzt. Alle üblichen Symbolimporte können Sie nutzen.
- Diverse Geräte die das **Modbus TCP** Protokoll einsetzen.
  - Modicon
  - Schneider
  - Wago
  - Beckhoff
  - Phoenix Contact
  - Omron
  - B&R
  - Fanuc
  - ABB
- **MQTT** Version 3 und 5. Eine einfache Itemsyntax öffnet die IoT Welt für OPC.
- IEC 60870-5-104. Das wird oft im Bereich der Versorgungen für Fernleitungen von Elektrizität, Wasser, Öl und Gas benutzt.
- Alle Systeme und Geräte die über **OPC UA** oder **Classic OPC** angesprochen werden könnenn. OPC Server und Client sind vorhanden.
- Raw data. Das sind reine Daten die keinem bestimmten Format unterliegen.

Kommuniziert über Ethernet

## BACnet

BACnet wird über IP / UDP genutzt.
Maximale Länge von Zeichenketten: 256 Byte
Unterstützte Zeichensätze: UTF-8, UTF-16, Latin-1
Statustexte werden unterstützt (state_text)
Unions ("Choice") und Strukturen ("Sequence") sind für wichtige Elemente wie Trend, Shedule, Calendar, Prioriry vorhanden. Trenddaten werden als History Daten angeboten. Alle nicht implementierten werden nicht angeboten.
Emum Werte werden standardmäßig aus UINT32 angeboten. Einige spezielle Enum werden auch als Bool dargestellt.
Werte der Typen "Octet-String" und "Bit-String" können nur als Ganzes geschrieben werden, kein Schreiben von Einzelelementen ist möglich.

### BBMD (BACnet Broadcast Management Device) Details

BBMD dient dem Verbindungsaufbau und der Gerätesuche wenn die Stationen nicht in der gleichen Kollisionsdomäne liegen. BACnet nutzt Broadcast zum Verbindungsaufbau. In umfangreicheren Umgebungen mit Unternetzen verwaltet BBMD diese Broadcasts.
Es gibt mehrere Verfahren BBMD zu nutzen, alle werden unterstützt:

- Suche mit Broadcast.
- Suche über die IP Geräteadresse.
- Suche über das BACnet ID.

Zusätzlich kann BBMD genutzt werden um ältere serielle Installationen an IP Netze anzukoppeln.

### COV (Change Of Values) Details

COV ist das Evenetsystem von BACnet. Events werden beim Browsen angeboten, sie werden abonniert. Sendet das Gerät die Daten so wird das Event gesendet.
Da BACnet mit UDP arbeitet kann der COV Empfang nicht garantiert werden. Deshalb bietet Tani optional eine Überwachung an. Die nutzt das Wiederverbindungs Timeout der Verbindung. Wird in der angegebenen Zeit kein Event empfangen so wird das Element aktiv nachgefragt. Hat sich der Wert dieser Nachfrage nicht geändert so wird kein Event ausgelöst.

### BACnet - Writing values with priority-array

These object types have a priority-array in addition to their present-value property:

- analog-output
- analog-value
- binary-output
- binary-value
- multi-state-output
- multi-state-value
- access-door

The BACnet spec says:

- priority-array is read-only and contains 16 entries (that can be a valid value or NULL).
- present-value is read-write and contains 1 value (the non-NULL value with the lowest priority from priority-array, or the value from relinquish-default if no non-NULL value in priority-array exists).
- Writing to present-value uses an optional priority parameter to write to the correct entry in priority-array.

The Tani implementation works as follows:

- priority-array is read-write and contains 16 structure entries with 2 fields:
  * Value: the data value in this entry (or 0 if no valid value is present)
  * ValueValid: a boolean value; 1 if Value is valid, 0 if not (NULL value).
- Writing to an element of priority-array implicitly uses a "write present-value with priority" operation to change the desired value.
- Writing to priority-array[i].Value always creates a non-NULL entry.
- Writing 0 to priority-array[i].ValueValid creates a NULL entry.
- Writing 1 to priority-array[i].ValueValid creates a non-NULL entry with value 0 (this is usually not very useful).
- Writing to priority-array[i] (as a structured data type) creates a NULL entry when ValueValid is 0. Else a non-NULL entry with the specified Value is created.
- present-value is read-write and contains the value obtained by BACnet protocol.
- Writing to present-value doesn't transfer the priority parameter. The BACnet device will implicitly write to priority entry 16 in this case.

This mechanism was chosen to allow choosing the write priority via OPC without changing the read syntax for present-value property. This also allows writing NULL values via OPC.

### Implemented Properties

The following object properties are implemented:

| Object Type | Property | BACnet Type | OPC Type | Remarks |
|---|---|---|---|---|
| all | all | BACnetObjectIdentifier | UInt32 | |
| all | all | Bit String | Array of Boolean | |
| all | all | Boolean | Boolean | |
| all | all | Character String | String | |
| all | all | Double | Double | |
| all | all | Enumerated | UInt32 | |
| all | all | Octet String | Array of UInt8 | |
| all | all | Real | Float | |
| all | all | Signed | Int32 | |
| all | all | Unsigned | UInt32 | |
| all | Change of State Time (16) | BACnetDateTime | DateTime | |
| all | Event Time Stamps (130) | Sequence of BACnetTimeStamp | Array of Structure "Timestamp" | |
| all | Object Type (79) | BACnetObjectType | UInt32 | |
| all | Time of Active Time Reset (114) | BACnetDateTime | DateTime | |
| all | Time of State Count Reset (115) | BACnetDateTime | DateTime | |
| Access Door (30) | Door Alarm State (226) | BACnetDoorAlarmState | UInt32 | |
| Access Door (30) | Present Value (85) | BACnetDoorValue | UInt32 | |
| Access Door (30) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "UnsignedPriorityValue" | see section "Priority Array" |
| Access Door (30) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Analog Input (0) | Present Value (85) | Real | Float | |
| Analog Input (0) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Analog Output (1) | Present Value (85) | Real | Float | |
| Analog Output (1) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "AnalogPriorityValue" | see section "Priority Array" |
| Analog Output (1) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Analog Value (2) | Present Value (85) | Real | Float | |
| Analog Value (2) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "AnalogPriorityValue" | see section "Priority Array" |
| Analog Value (2) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Averaging (18) | Maximum Value Timestamp (149) | BACnetDateTime | DateTime | |
| Averaging (18) | Minimum Value Timestamp (150) | BACnetDateTime | DateTime | |
| Binary Input (3) | Present Value (85) | BACnetBinaryPV | UInt32 | |
| Binary Input (3) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Binary Output (4) | Present Value (85) | BACnetBinaryPV | UInt32 | |
| Binary Output (4) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "UnsignedPriorityValue" | see section "Priority Array" |
| Binary Output (4) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |

| Object Type | Property | BACnet Type | OPC Type | Remarks |
|---|---|---|---|---|
| Binary Value (5) | Present Value (85) | BACnetBinaryPV | UInt32 | |
| Binary Value (5) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "UnsignedPriorityValue" | see section "Priority Array" |
| Binary Value (5) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Calendar (6) | Datelist (23) | List of BACnetCalendarEntry | Array() of Structure "BACnet.CalendarEntry" | |
| Device (8) | Last Restore Time (87) | BACnetTimeStamp | Structure "Timestamp" | |
| Device (8) | Local Date (56) | Date | Structure "Date" | |
| Device (8) | Local Time (57) | Time | Structure "Time" | |
| Device (8) | Object List (76) | Sequence of BACnetObjectIdentifier | Array of UInt32 | |
| Device (8) | Protocol Object Types Supported (96) | BACnetObjectTypesSupported | Array of Boolean | |
| Device (8) | Protocol Services Supported (97) | BACnetServicesSupported | Array of Boolean | |
| Device (8) | Segmentation Supported (107) | BACnetSegmentation | UInt32 | |
| Device (8) | System Status (112) | BACnetDeviceStatus | UInt32 | |
| Device (8) | Time of Device Restart (203) | BACnetTimeStamp | Structure "Timestamp" | |
| Event Enrollment (9) | Object Property Reference (78) | BACnetDeviceObjectPropertyReference | Structure "DeviceObjectPropertyReference" | |
| File (10) | Modification Date (149) | BACnetDateTime | DateTime | |
| Life Safety Point (21) | Present Value (85) | BACnetLifeSafetyState | UInt32 | |
| Life Safety Point (21) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Life Safety Zone (22) | Present Value (85) | BACnetLifeSafetyState | UInt32 | |
| Life Safety Zone (22) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Load Control (28) | Actual Shed Level (212) | BACnetShedLevel | Structure "ShedLevel" | |
| Load Control (28) | Duty Window (213) | Unsigned | UInt32 | |
| Load Control (28) | Expected Shed Level (214) | BACnetShedLevel | Structure "ShedLevel" | |
| Load Control (28) | Present Value (85) | BACnetShedState | UInt32 | |
| Load Control (28) | Requested Shed Level (218) | BACnetShedLevel | Structure "ShedLevel" | |
| Load Control (28) | Shed Duration (219) | Unsigned | UInt32 | |
| Load Control (28) | Start Time (142) | BACnetDateTime | DateTime | |
| Loop (12) | Controlled Variable Reference (19) | BACnetDeviceObjectPropertyReference | Structure "DeviceObjectPropertyReference" | |
| Loop (12) | Manipulated Variable Reference (60) | BACnetDeviceObjectPropertyReference | Structure "DeviceObjectPropertyReference" | |
| Loop (12) | Setpoint Reference (109) | BACnetSetpointReference | Structure "SetpointReference" | |
| Loop (12) | Present Value (85) | Real | Float | |
| Loop (12) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Multi State Input (13) | Alarm Values (7) | Sequence of Unsigned | Array of UInt32 | |
| Multi State Input (13) | Fault Values (39) | Sequence of Unsigned | Array of UInt32 | |
| Multi State Input (13) | Present Value (85) | Unsigned | UInt32 | |
| Multi State Input (13) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Multi State Output (14) | Present Value (85) | Unsigned | UInt32 | |
| Multi State Output (14) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "UnsignedPriorityValue" | see section "Priority Array" |
| Multi State Output (14) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Multi State Value (19) | Alarm Values (7) | Sequence of Unsigned | Array of UInt32 | |
| Multi State Value (19) | Fault Values (39) | Sequence of Unsigned | Array of UInt32 | |
| Multi State Value (19) | Present Value (85) | Unsigned | UInt32 | |
| Multi State Value (19) | Priority Array (87) | BACnetPriorityArray | Array(1..16) of Structure "UnsignedPriorityValue" | see section "Priority Array" |
| Multi State Value (19) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Notification Class (15) | Recipient List (102) | List of BACnetDestination | Array() of Structure "BACnet.Destination" | |

| Object Type | Property | BACnet Type | OPC Type | Remarks |
|---|---|---|---|---|
| Schedule (17) | Effective Period (32) | BACnetDateRange | Structure "DateRange" | |
| Schedule (17) | Exception Schedule (38) | Sequence of BACnetSpecialEvent | Array of Structure "SpecialEvent" | |
| Schedule (17) | List of Object Property References (54) | Sequence of BACnetDeviceObjectPropertyReference | Array of Structure "DeviceObjectPropertyReference" | |
| Schedule (17) | Present Value (85) | ABSTRACT-SYNTAX.&Type | Structure "Any" | |
| Schedule (17) | Schedule Default (174) | ABSTRACT-SYNTAX.&Type | Structure "Any" | |
| Schedule (17) | Weekly Schedule (123) | Sequence Size(7) Of BACnetDailySchedule | 7 sub-objects ("Monday", "Tuesday", ...) of Structure "TimeValue" | |
| Pulse Converter (24) | Present Value (85) | Real | Float | |
| Pulse Converter (24) | Status Flags (111) | BACnetStatusFlags | Array(0..3) of Boolean | |
| Structured View (29) | Subordinate List (211) | Sequence of BACnetDeviceObjectReference | Array of Structure "DeviceObjectReference" | |
| Trend Log (20) | Client COV Increment (127) | BACnetClientCov | Structure "ClientCov" | |
| Trend Log (20) | Log Buffer (131) | BACnetLogRecord | Structure "LogRecord" | Accessed via "HistoryRead" function, "Read" shows only one record. |
| Trend Log (20) | Log Device Object Property (132) | BACnetDeviceObjectPropertyReference | Structure "DeviceObjectPropertyReference" | |
| Trend Log (20) | Start Time (142) | BACnetDateTime | DateTime | |
| Trend Log (20) | Stop Time (143) | BACnetDateTime | DateTime | |

## KNX and EIB

KNX wird für IP / TCP und IP / UDP angeboten.
Der Symbolimport wird für die standardisierten ESF Dateien angeboten.

## Diagnoselogger

Der OPC Server enthält einen Logger zur Diagnose bei Inbetriebnahmen. Dieser Logger kann konfiguriert werden. Werden in großen Anlagen alle SPS Daten geloggt so kann die Rechnerbelastung groß werden.

## Grenzen

Maximale Anzahl konfigurierbarer Client Verbindungen: 4000.
Maximale Datenelementlänge eines Elements: 4GB.
Maximale Anzahl Elemente je Verbindung: 1 Million.
Maximale Anzahl der Summe aller Elemente (Items): 16 Millionen.
Maximale OPC Gruppen je Verbindung: 100.
Maximale Anzahl Verbindungen je passivem Port ist 999.
Die OPC synchronen Funktionen liefern direkt einen Fehler wenn die SPS Verbindung nicht steht.
Änderungen in der SPS Konfiguration werden alle 10 Sekunden geprüft falls die Steuerung keine Überprüfung bei jedem Datenschreiben anbietet.
Felder können jedes bis zu 64Kb lang sein.
Mehrdimenionale Felder können bis zu sechs Dimensionen haben.

## Verbindungssimulation

Verbindungen können simuliert werden.
Beim Lesen werden die Elemente angelegt mit Wert 0. Ein Schreiben ändert Werte, diese Werte werden beim Lesen wieder zurückgegeben.
Grenzen sind:

- Strings können maximal mit 16 Zeichen beschrieben werden
- Suffixe werden nicht abgearbeitet
- Datum und Uhrzeitelemente werden nur eingeschränkt unterstützt

## Virtuelle Verbindungen

Eine virtuelle Verbindung wird überwiegend als Ziel von Variablenunlenkungen genutzt.
Jede virtuelle Verbindung hat eine individuelle Rechnreeinstellung wie alle anderen Verbindungen ebenfalls. Wie bei den Logiktabellen und der Status Variable Funktion wird das Recht der Quellverbindung nicht beachtet.
Symbole werden über den Symboleditor bearbeitet.
Ein und das selbe SPS Element darf in einer virtuellen verbindung nur einmal benutzt werden.

Je nach Lizenz können weniger Elemente oder Längen möglich sein.

## Geschwindigkeit

Der Datendurchsatz wird von der SPS Geschwindigkeit oder der Reaktionszeit der Applikation die OPC nutzt bestimmt.
Leseanfragen an die SPS werden so weit die SPS das zulässt optimiert. Dazu werden die angeforderten Elemente zu Blöcken zusammengefasst wenn das geht - bei Ein und Ausgängen geht das nicht. Diese Optimierung kann abgeschaltet werden, ebenso kann pro SPS Verbindung dieses Datenzusammenfassen eingestellt werden.
Schreibanfragen werden je nach Konfiguration entweder zusammengefasst, oder die Reihenfolge der Schreibaufträge wird so eingehalten wie die OPC Client Applikation schreibt.
An den OPC Schnittstellen werden alle Optimierungen genutzt die die jeweilige OPC Schnittstelle kennt.
Die normale Zeit beim zyklischen Anfragen der SPS beträgt 50ms. Sie ist schneller wenn das Steuerungs Abfrageintervall auf 0 gesetzt wird
Zu OPC werden nur Daten gesendet die sich in der SPS zwischen zwei Leseanfragen geändert haben.

### Feld und Textoptimierungen

Die ab Version 1.8 vorhandenen OPC Feldoptimierungen verhindern das zu häufige Lesen der langen Datenfelder, nur der Index wird zyklisch abgefragt. Diese Optimierung hilft nur dann wenn sich der Index nicht allzuoft ändert.

## Speicherbedarf

- Programmcode: Minimal werden 6MB für den Programmcode benötigt. Der genaue Speicherbedarf hängt stark von Arbeitsweisen der Betriebssysteme ab. So werden Bibliotheken normalerweise nur einmal geladen. Wird z.B. die Standardbibliothek schon von einem anderen laufenden Programm genutzt so ist sie schon geladen, andernfalls werden weitere ca. 4MB Speicher belegt.
- Nutzdaten: Minimal belegt der OPC Server 2MB Daten intern. Zusätzlich werden die Steuerungsdaten intern in einem Zwischenspeicher gehalten, der dient Alt Neuvergleichen. Der Zwischenspeicher belegt je Item die Datenlänge des Items + 64 Bytes. Jede Verbindung belegt 4kB Speicher.

## Rechenzeitbedarf

Die genutzte CPU Zeit hängt von der Auslastung ab. Meist wird auf die Daten der Steuerungen oder Daten von der OPC Applikation gewartet.
Alle Software arbeitet intern ereignisgesteuert. Das dient dem Durchsatz und minimiert die CPU Belastung.
Mehrere CPU im PC werden unterstützt. Maximal werden 10 CPU genutzt, die Haupt Verarbeitungslast wird von drei CPUs geleistet.

## Installation

Die Installation installiert je nach Produkt und Auswahl mehrere Programmteile. Beim Uninstall werden nicht automatisch alle Teile gelöscht. Alle Uninstaller werden aber im Menü und unter Software angeboten, gegebenenfalls müssen die Produkte einzeln entfernt werden.
Die Anwender Einstellungen werden beim Uninstall nicht gelöscht.

## Automatic structure import

Type Auto-Import works for all client protocols that are able to use structures/enumerations and have online browsing functions. This includes:

- OPC UA
- OpcPipe
- Siemens S7-1500
- Rockwell CompactLogix/ControlLogix/MicroLogix
- IEC104
- KNX

These protocols have a fixed list of structures and don't need Auto-Import:

- BACnet

These protocols have online browsing, but don't use structures/enumerations:

- OPC DA
- MQTT

All other protocols don't have online browsing.

Type Auto-Import is implemented in two steps:

1. A structure or enumeration type which has not been imported is assigned a Node ID when:
    - the Item is being monitored (by calling CreateMonitoredItems):
    - the Item is being read/written (by calling Read/Write):
    - the Item is being registered (by calling RegisterNodes):
    - the DataType attribute of an Item with this type is accessed:
2. A structure or enumeration type which has not been imported is actually imported when:
    - the Item is being monitored (by calling CreateMonitoredItems):
    - the Item is being read/written (by calling Read/Write):
    - the Item is being registered (by calling RegisterNodes):
    - the DataTypeDefinition attribute of the DataType node is read (after it has been created by step 1):
    - the EnumValues property node is read (for Enumerations, after it has been created by step 1):

Limitations:

Before Auto-Import Step 1, any types that have not been imported yet:

- are not available anywhere

Before Auto-Import Step 2, any types that have not been imported yet:

- have a DataType Node ID assigned
- are not browseable in Types/DataTypes/BaseDataType/Structure/UserStructures or Types/DataTypes/BaseDataType/Enumeration
- are not present in the XML data in Types/DataTypes/OPC Binary/UserStructures
- are not browseable in Types/VariableTypes/BaseVariableType/BaseDataVariableType/UserStructures
- don't have the type comment available
- don't have Encoding Node IDs available

After Auto-Import Step 2:

- the newly imported types behaves exactly as any manually imported type
- if the type later changes in the source systen, the import cache will NOT be updated

A client wishing to use a variable with a structure/enumeration type that has not been imported should

- either read the DataType attribute of the variable, then read the DataTypeDefinition attribute/EnumValues property of the type node,
- or monitor/read the Value attribute of the variable before checking the data types

to trigger the type import. Only after completing one of these the structure type is available in the server.

## Betriebssysteme

- Windows 10, 11 (alle Versionen), 64 Bit. Ältere wie Vista oder Windows 7, 8 64 und 32 Bit liegen als letzte stabile Version bereit.
- Windows Server 2008, 2012, 2016, 2019 und 20122.
- Windows XP 32 Bit. Das ist ein Spezialstand mit dem OPC Server Kern.
- Linux auf dem Raspberry Pi und Odroid Computer (64 und 32 Bit).
- Linux auf Phytec Geräten wie den Regor, Tauri S (32 Bit) und Tauri L (64 Bit).
- Linux auf der Wiesemann & Theis pure.box und pure.box5.
- Linux am PC: Debian, Ubuntu, Suse, Arch, Centos, Redhat und weitere Distributionen.
- Linux 64 Bit als Docker or Kubernetes Container.

- OPC DA erfordert Microsoft Windows. Es werden alle von Microsoft unterstützten Intel Betriebssysteme und Anwendersprachen unterstützt. Es muss das aktuelle Service Pack installiert sein. OPC DA erfordert mindestens 2 CPU im Rechner um eine gute Performance zu erreichen.

- Unter Windows arbeitet der OPC Server als Dienst, unter Linux als Daemon.
- Die Raspberry Version unterstützt alle Linux Distributionen für diese Platform.
- Alles Andere arbeitet unter unterschiedlichen Systemen, oft Linux kompatibel.
- Unter Linux wird für den OPC Server ein POSIX kompatibles System vorausgesetzt. Die Standard Library muß mindestens V2.2 haben. Die Konfigurationssoftware benötigt die KDE 5 Libraries. Nutzen Sie marktgängige Distributionen wie Debian, Ubuntu, Suse, Redhat oder ähnlich.
- Getestet ist: Windows Intel 32 und 64 Bit, Linux Intel 32 und 64 Bit, Linux MIPS 32 CPU, Linux ARM 32 und 64 Bit CPU.
- Der Einsatz in virtuellen Maschinen wird unterstützt. Docker und Kubernetes Container werden ebenfalls unterstützt.
- Bei Windows7 muss mindestens Service Pack 1 vorhanden sein um H1 nutzen zu können. Alle Windows service packs für SHA512 müssen installiert sein.

- Alle Konfigurationen sind kompatibel zu allen OPC Servern, auch über Betriebssystemgrenzen hinweg.