

OPC Server for Johnson Controls N2 1.02 User's Manual



Mark "Johnson Controls" and mark "Metasys" are owned by Johnson Controls, Inc. and/or its affiliates.

OPC Labs, s.r.o. is not associated with Johnson Controls. Johnson Controls has not provided any assistance in developing this product, and have not made determination that the server is compatible with Johnson Controls' products.

Contents

Introduction	6
Block Diagrams	7
Connection via serial port	7
Connection using TCP through an Ethernet/serial convertor.....	8
Modes of Operation	8
Installation	9
Operating Systems.....	10
Prerequisites.....	10
Licensing	11
Related Products	12
Product Parts	13
OPC Server Runtime	13
Configuration Application	13
Utilities	13
License Manager.....	13
Documentation and Help	14
Fundamentals	15
Configuration Objects.....	15
Hardware Types	15
N2 Hardware Type.....	16
N2Open Hardware Type.....	17
Modules and Module Types	17
Objects and Object Types	18
Channels.....	18
Serial Port	18
Socket	20
Controllers	20
N2 Master Controller.....	21
Optomux Forwarder	21
Devices	22
N2 Device.....	22
N2Open Device.....	23
Items	23
N2 Object Item	24
N2 Functional Module Item	25
N2Open Field Item	25



N2Open Command Item	26
OPC Address Space.....	27
Names and Data IDs.....	30
Device Branches.....	30
Device Items.....	30
Module Type Branches	30
Module Branches	31
Module Items.....	31
Properties.....	31
OPC Operations	32
Reading/writing BCD Data in N2.....	32
Commands in N2Open.....	32
Configuration.....	34
Overview.....	34
Panels	34
Document Panel.....	34
Node List Panel	35
Edit Values Panel.....	36
Properties Panel.....	37
Command Groups.....	37
File Commands.....	37
Edit Commands	37
View Commands	38
Tools Commands.....	38
Help Commands.....	39
Operations.....	40
OPC Client Connections.....	40
Server Control.....	40
Troubleshooting	41
Error Codes	41
Advanced Topics	44
OPC Specifications.....	44
OPC-UA (Universal Architecture)	44
Configuring the Server for OPC-UA.....	45
OPC Interoperability.....	46
Event Logging	46
N2 OPC Settings Utility	47
Component Parameters Page.....	47
Address Space Group	48

Module Parameters Page	49
Server Report Group	49
System Parameters Page	49
Module Parameters Group.....	50
COM Registration and Server Types.....	51
Internal Optimizations.....	51
Failure Recovery	51
Protocol Errors	52
Transient Communication Errors.....	52
Channel Errors	52
Data Types	53
Multithreading and Synchronization.....	53
Additional Resources	54
Appendix A. Optomux, N2 and N2Open Error Codes	55
Optomux Error Codes.....	55
N2 Error Codes	55
N2Open Error Codes.....	56

Introduction

The OPC server acts as a master on Johnson Controls Metasys N2 network consisting of a mix of slave devices communicating using N2 (original Johnson Controls protocol) or N2Open protocol. N2 and N2Open are Optomux-based protocols used in HVAC (Heating, Ventilation and Air Conditioning) and other building automation networks. The server supports DC, DX, TC, VND and many other N2/N2Open devices, and is extensible using “hardware type definition” files.

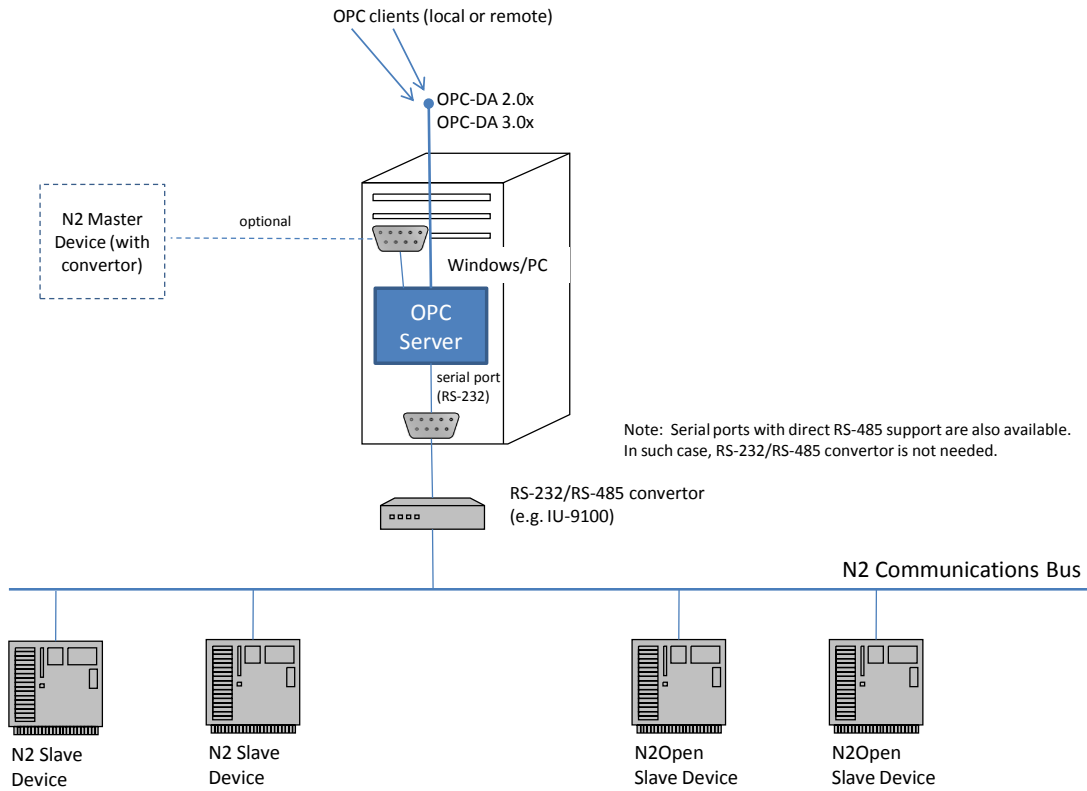
Can be used to replace the NCM (Network Control Module) and enable OPC access to N2 slaves. Optionally, the OPC server can also pass through (forward) communication from other master, such as the NCM unit.

The server can handle multiple N2 networks simultaneously, and up to 255 slave devices on each network. The computer can connect to an N2 network via serial port, or using TCP through an Ethernet/serial convertor.

Configuration program with rich user interface is supplied with the server. You can use both the items pre-defined for a hardware type of your device, or add your own named items. The configuration is stored in an XML file. The server can run as Windows Service or Local Server.

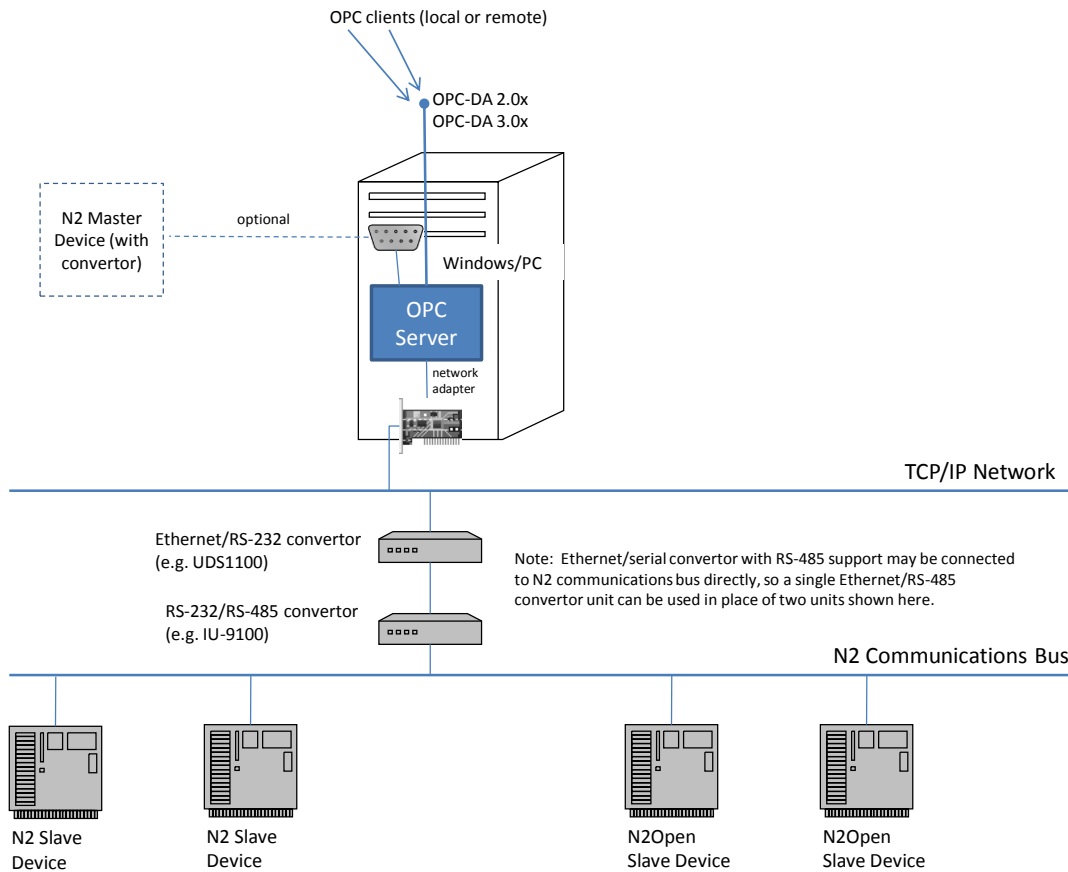
Block Diagrams

Connection via serial port



Serial port may be internal, or external to the PC (e.g. connected by USB).

Connection using TCP through an Ethernet/serial convertor



Modes of Operation

The OPC server collects data by reading values from N2 and N2Open slaves. The reading is one-shot (for OPC reads) and/or periodic (for OPC subscriptions). Multiple subscriptions to the same item are merged together.

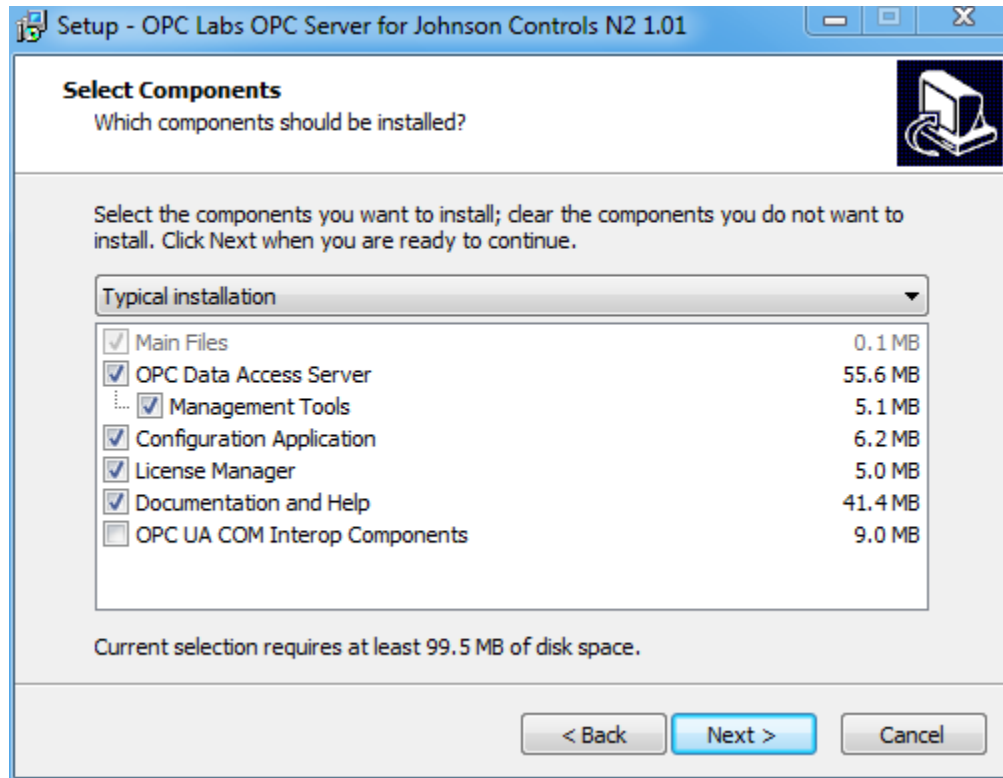
OPC writes are transformed to N2 write commands on per call basis.

Installation

The installation can be started by running the setup program. Just follow the on-screen instructions. The installation program requires that you have administrative privileges to the system.



The installation program offers several installation types, and allows you to choose specifically which part of the product to install. For start, you can simply keep the defaults.



After the installation is finished, you can access the documentation and various tools from your Start menu.

The product includes an uninstall utility and registers itself as an installed application. It can therefore be removed easily from Control Panel. Alternatively, you can also use the Uninstall icon located in the product's group in the Start menu.

Operating Systems

The product is supported on following operating systems:

- Microsoft Windows XP with Service Pack 2 or later (x86)
- Microsoft Windows Vista with Service Pack 1 or later (x86, x64)
- Microsoft Windows 7 (x86, x64)
- Microsoft Windows Server 2003 with Service Pack 1 or later (x86)
- Microsoft Windows Server 2008 (x86, x64)
- Microsoft Windows Server 2008 R2 (x64)

Prerequisites

The following software must be present on the target system before the installation:

1. Microsoft .NET Framework 2.0 with Service Pack 1 (x86). You can install it from <http://www.microsoft.com/downloads/details.aspx?FamilyId=79BC3B77-E02C-4AD3-AACF-A7633F706BA5>.

The N2 OPC server does not directly require the Microsoft .NET Framework 2.0, but OPC Core Components setup may fail without it. If Microsoft .NET Framework 3.5 SP1 is installed (see below), Microsoft .NET Framework 2.0 is not needed.

2. Microsoft .NET Framework 3.5 with Service Pack 1 (x86). You can install it from <http://www.microsoft.com/downloads/details.aspx?FamilyID=ab99342f-5d1a-413d-8319-81da479ab0d7>.

This is version of the Microsoft .NET Framework that is used by the OPC server (for the configuration application). It is also required if you are going to use OPC Universal Architecture (OPC UA COM Interop Components).

The setup program also installs following software on the target system:

1. Microsoft Core XML Services (MSXML) 6.0 with Service Pack 1 (x86)
2. Microsoft Visual C++ 2010 Redistributable Package (x86)
3. OPC Core Components 3.00 Redistributable (x86 or x64)

Licensing

The OPC Server for Johnson Controls N2 is a licensed product. You must obtain a license to use it in development or production environment. For evaluation purposes, you are granted a trial license, which is in effect if no other license is available.

With the trial license, the OPC server only provides valid OPC data for 30 minutes since the OPC server was started. After this period elapses, the server will no longer provide data reading and write through OPC. Restarting the OPC server gives you additional 30 minutes, and so on. If you need to evaluate the product but the default trial license is not sufficient for your purposes, please contact the vendor or producer, describe your needs, and a special trial license may be provided to you.

The limitations of the trial license only apply to the runtime party of the OPC server and its ability to read and write data through OPC. All other features of the software, such as the configuration tool, remain fully functional.

The licenses are installed and managed using a License Manager utility, described further in this document.

Related Products

Additional products exist to complement the base N2 OPC server offering. For instance, you may use the QuickOPC-COM or QuickOPC.NET product if you want to develop own applications that integrate OPC client functionality. Check the options available with your vendor.

Product Parts

OPC Server Runtime

This is an executable that implements the OPC Server's COM objects. Besides notifications that may appear in the system tray area of the taskbar, the runtime has no user interface on itself. The COM/DCOM infrastructure takes care of starting the runtime when necessary; alternatively, you can also start it manually.

The runtime reads its configuration information upon startup from the configuration XML file, usually created by the user with the help of configuration application.

Configuration Application

Configuration application is a program that is separate from the OPC server's runtime part. It provides user interface for creating, viewing and modifying of the information that the OPC server uses to communicate with the devices, and provide data to OPC clients.

Configuration information is stored in an XML file with pre-defined structure (schema). The configuration application can open and work with any XML file that conforms to the schema. Only one such file, however, is marked as active for the runtime part, and this is the one that the runtime will load and execute.

Utilities

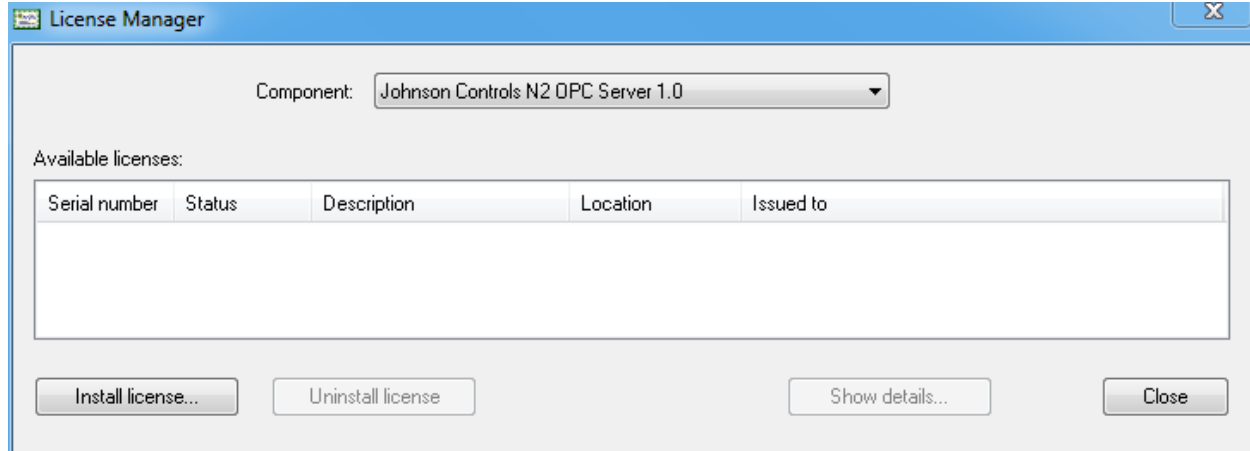
The N2 OPC Settings utility allows to view and edit various advanced settings that influence how the runtime part of the OPC server works.

The Event Log Options utility allows you to influence how the OPC server logs relevant internal events, and which groups of event get logged.

License Manager

The License Manager is a utility that allows you to install, view and uninstall licenses.

In order to install a license, invoke the License Manager application (from the Start menu), and press the **Install license** button. Then, point the standard Open File dialog to the license file (with .BIN) extension provided to you by your vendor.



Documentation and Help

The documentation consists of following parts:

- User's Manual (this document).
- OPC Server for Johnson Controls N2 Datasheet.
- Event Log Options Help. Describes the Event Log Options utility.

You can access the above mentioned documentation either from the Start menu, or from the respective applications.

Fundamentals

Configuration Objects

The configuration consists of various objects, arranged in a hierarchical (tree-like) structure. In addition, the configuration contains references (links) between objects that exist in otherwise unrelated parts of the tree.

The overall structure (schema) of the configuration is given by the software, but the actual contents is up to you. The schema defines which types of objects exist, what are their attributes, and how can objects be composed together.

There are many different types of configuration objects, and their description is given further below. In order to gain some basic understanding of the configuration structure, please study the following summary:

- The *hardware types* define types of N2 or N2Open devices that the OPC server can communicate to.
- The *modules*, *module types*, *objects*, and *object types* are used to structure the data in the hardware type definitions.
- The *channels* define the serial ports and TCP/IP connections that the OPC servers uses to communicate with the devices.
- The *controllers* are active entities within the OPC server that act as masters or slaves on N2/N2Open communication bus. There should be one controller under each channel.
- The *devices* are the actual physical units connected to channels. Each device is a slave on N2/N2Open communication bus. Each device can have a hardware type associated with it, and is residing on certain channel.
- The *items* are addressable data elements in the device, made accessible by the OPC server to the OPC clients.

The root of the configuration contains three pre-defined containers: for hardware types, channels, and devices. The controllers are defined under the channels, and the items are defined under the devices.

Hardware Types

The *hardware types* define types of N2 or N2Open devices that the OPC server can communicate to.

Each hardware type is given a unique name. For a device in your configuration, you can specify its hardware type name, and the device will assume characteristics and contents (such as objects and data items) of this type.

An external hardware type is defined in a file separate from the configuration. In this version, all hardware types are external, and their definition files reside in a dedicated directory. The definition file must be named after the hardware type specified in this part of configuration.

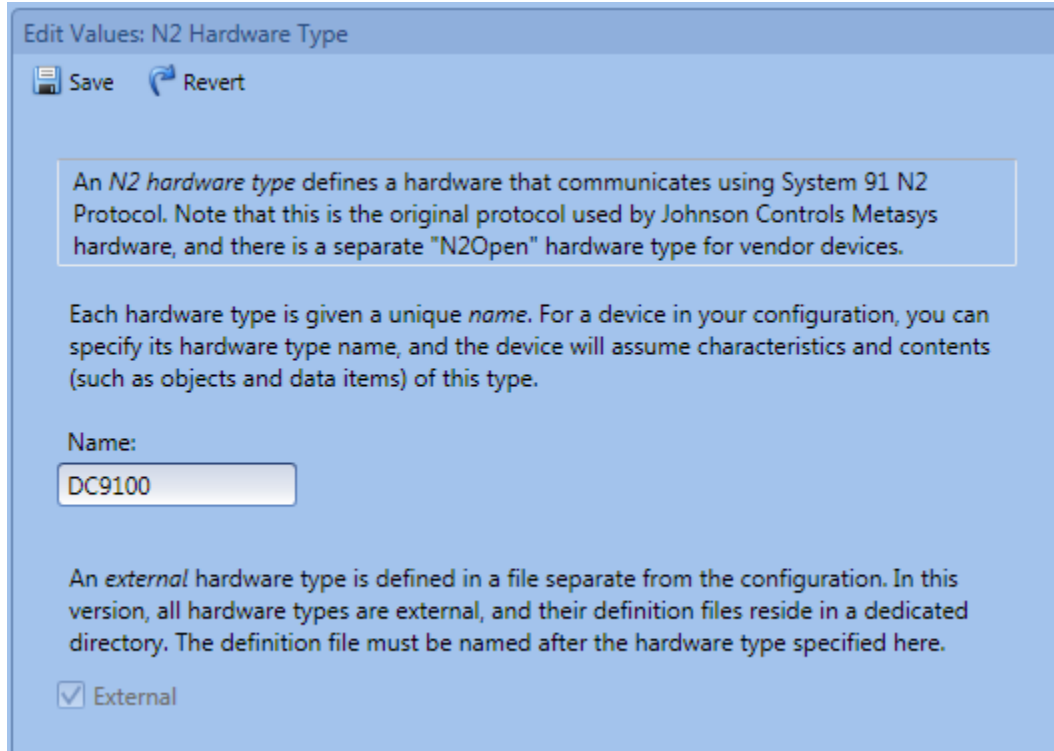
The OPC server comes with several pre-defined hardware types, such as DC9100, DX9100, TC9100, or VND. You can simply use these hardware types, by assuring that their names appear listed under the Hardware Types configuration node, and then referencing them from particular devices of that type.

You can also define new hardware types by supplying their definition files in XML format, and placing them in the folder alongside with the pre-defined types. For guidance with creating the hardware type definition files, contact your vendor.

The OPC server can also work with devices without specifying their hardware type, as long as the device is compatible with N2 or N2Open protocol, and you have information about addresses and other characteristics of the items you want to access. The disadvantage of this approach is that if you have multiple similar devices, the item definitions need to be repeated under each device. There are also some limitations to this approach, and some advanced features are not available with it. Specifically, without a hardware type, the OPC server currently only allows you to define basic N2 and N2Open items (N2 Object Items and N2Open Field Items), but the more advanced items, such as N2 Functional Module Items and N2Open Command Items. Bit-coded fields cannot currently be defined in this way, either.

N2 Hardware Type

An *N2 hardware type* defines a hardware that communicates using System 91 N2 Protocol. Note that this is the original protocol used by Johnson Controls Metasys hardware, and there is separate “N2Open” hardware type for vendor devices.



An example of N2 hardware type is 'DC9100'.

N2Open Hardware Type

An *N2Open hardware type* defines a hardware that communicates using N2Open protocol for vendor devices. Note that for devices that communicate using the original Johnson Controls System 91 N2 Protocol, there is a separate "N2" hardware type.

A typical example of N2Open hardware type is 'VND'.

Modules and Module Types

Modules and *module types* can currently only appear in the external hardware type definition files, not in the configuration application.

We use the term *module type* for a union that contains object types all related to a single functionality area. For example, one of module types in DX-9100 hardware is TS (Time Schedule Module). This module type comprises of two object types: "Time Scheduling Items", which is a regular object type for items accessed using normal N2 read/write messages, and "Time Schedule Module Messages", which is an object type for a functional module, whose data is accessed using a special messages for reading/writing functional modules blocks.

In many (most) cases though, there is a one-to-one correspondence between module type and its object type.

A *module* is an instance of module type. For example, AI1 (Analog Input Module #1) in DX-9100 hardware is an instance of AI (Analog Input Module) module type.

Objects and Object Types

Objects and *object types* can currently only appear in the external hardware type definition files, not in the configuration application.

The object type has a set of items that always exist for this type of object. Different classes of object types exist (such as those for regular N2 objects, N2 functional modules, and N2Open objects). Depending on the class of the object type, it may also have other associated properties, such as the N2Open region it refers to.

An *object* is an instance of object type. For example, BI1 (Binary Input #1) in VND hardware is an instance of BI (Binary Input) object type.

Channels

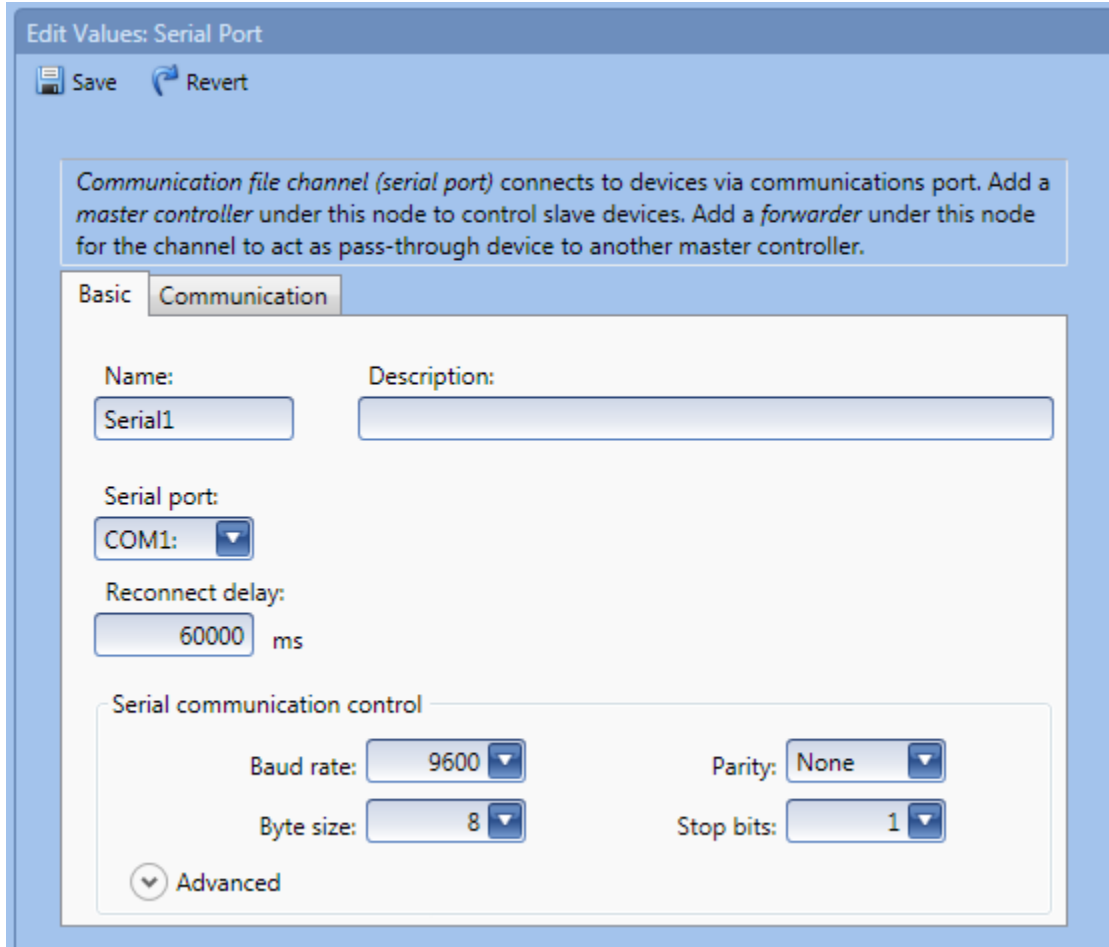
The *channels* define the serial ports and TCP/IP connections that the OPC servers uses to communicate with the devices.

Channels of following types can be defined under the Channels container, in any number or combination:

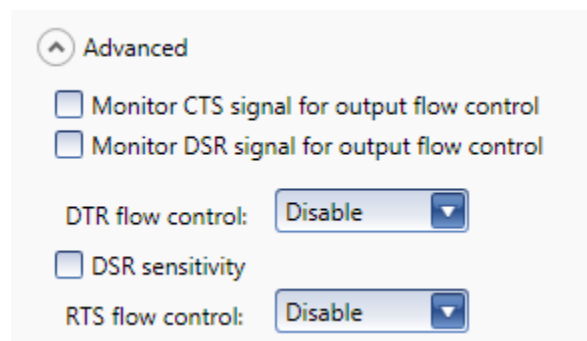
- Serial Port
- Socket

Serial Port

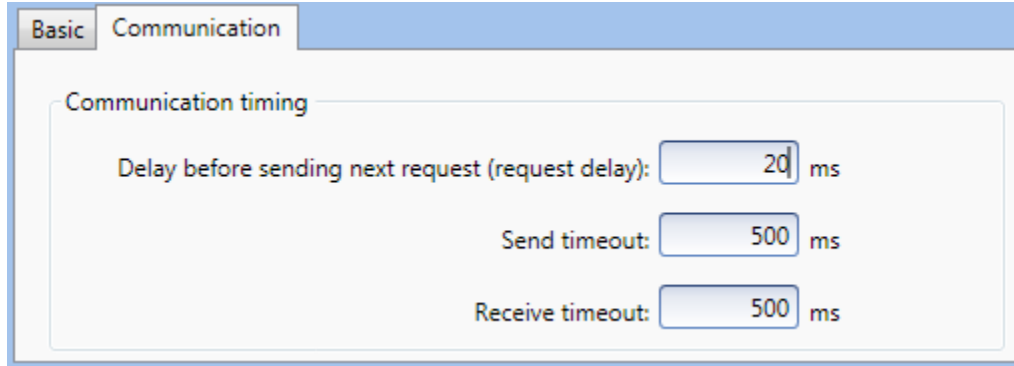
Communication file channel (serial port) connects to devices via communication port. Add a master controller under this node to control slave devices. Add a forwarder under this node for the channel to act as pass-through device to another master controller.



Expanding the Advanced part displays a set of controls mainly influencing the serial flow control features (below). In normal installations, there should be no need to change the flow control settings.



The Communication tab contains parameters that govern the message timing used on this channel.



Basic Communication

Communication timing

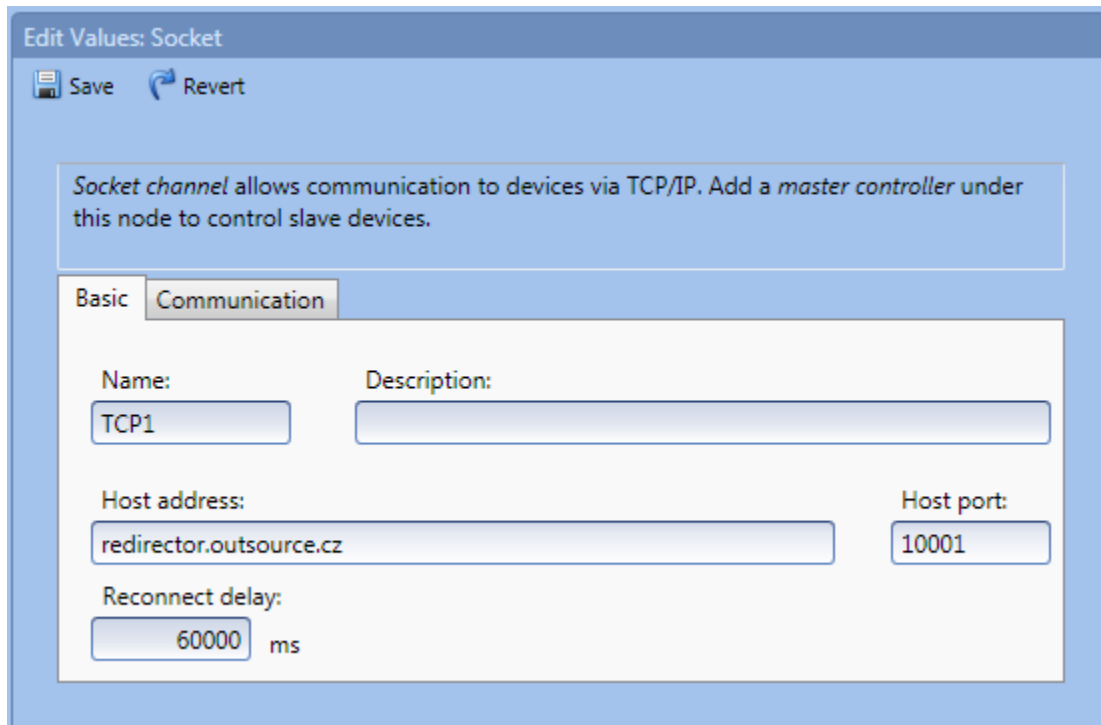
Delay before sending next request (request delay): ms

Send timeout: ms

Receive timeout: ms

Socket

Socket channel allows communication to devices via TCP/IP. Add a master controller under this node to control slave devices.



Edit Values: Socket

Save Revert

Socket channel allows communication to devices via TCP/IP. Add a master controller under this node to control slave devices.

Basic Communication

Name: Description:

Host address: Host port:

Reconnect delay: ms

The Communication tab is similar to the one in Serial Port. In addition, there is a Baud Rate field. While the computer (OPC server) does not set the baud rate of the socket channel (it has to be set on the Ethernet/serial convertor), the value configured here is necessary for proper computation of timing parameters for Optomux protocol, internally in the OPC server.

Controllers

The *controllers* are active entities within the OPC server that act as masters or slaves on N2/N2Open communication bus. There should be one controller under each channel.

You can add no more than one controller under each channel. Controllers of following types can be defined under the channel:

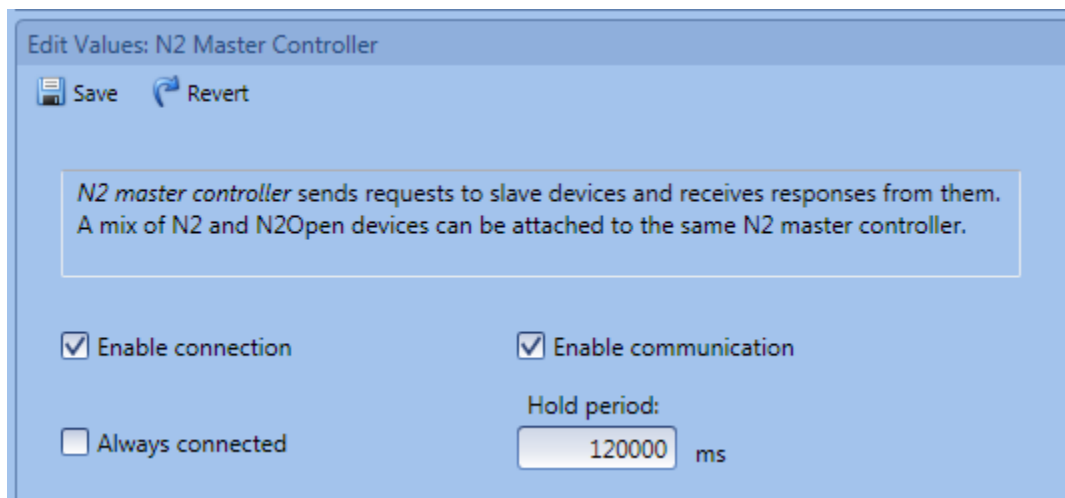
- N2 Master Controller
- Optomux Forwarder

A channel without a controller is allowed by the configuration application, but such channel does not serve any meaningful purpose.

Typically, you add N2 Master Controller under each channel. The Optomux Forwarder is only used in special cases.

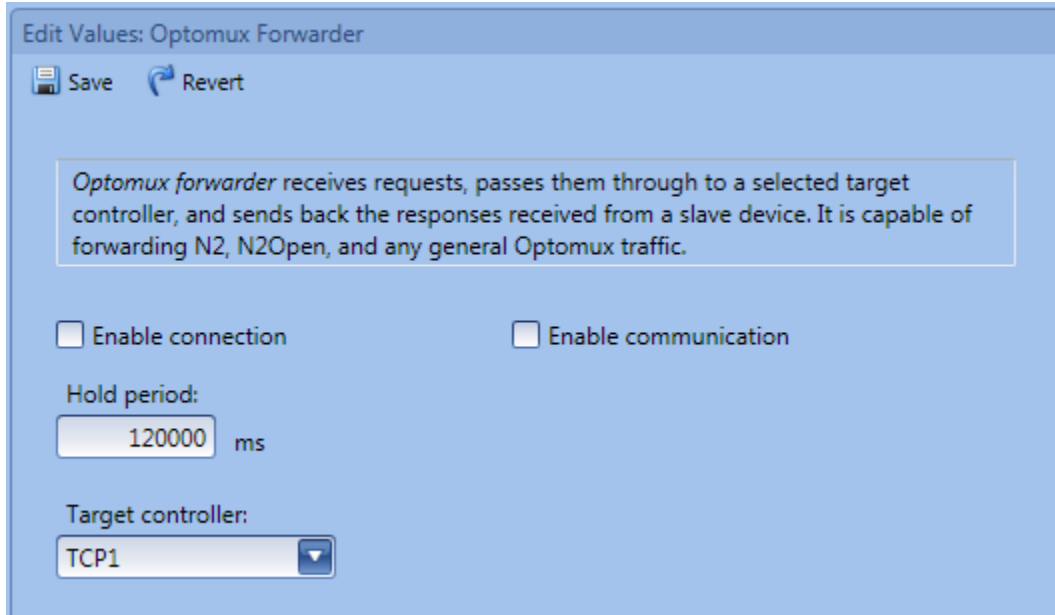
N2 Master Controller

N2 master controller sends requests to slave devices and receives responses from them. A mix of N2 and N2Open devices can be attached to the same N2 master controller.



Optomux Forwarder

Optomux forwarder receives requests, passes them through to a selected target controller, and sends back the responses received from a slave device. It is capable of forwarding N2, N2Open, and any general Optomux traffic.



Devices

The *devices* are the actual physical units connected to channels. Each device is a slave on N2/N2Open communication bus. Each device can have a hardware type associated with it, and is residing on certain channel.

Devices of following types can be defined under the Devices container, in any number or combination:

- N2 Device
- N2Open Device

N2 Device

N2 device is a unit that communicates using System 91 N2 protocol.

Note: For devices that use the N2Open protocol (such as vendor devices), use the “N2Open device” instead.

Edit Values: N2 Device

Save Revert

N2 device is a unit that communicates using System 91 N2 Protocol.

Note: For devices that use the N2Open protocol (such as vendor devices), use the "N2Open device" class instead.

Name: Device1 Enabled

Description: Development DX-9100

Channel name: TCP1 *Channel name determines the serial port or TCP/IP socket used to reach this device. Select one of the named channels defined in the configuration.*

Address: 1 *Address is a number that uniquely identifies the device on the bus. Unit addresses are from 1 to 255, address 0 is reserved.*

Hardware type: DX9100 *Hardware type defines characteristics and contents (such as pre-defined objects and data items) of the device. Select a named type, or leave the type empty.*

Retries before suspend: 3

Delay before resume: 10000 ms

Receive timeout: 500 ms

N2Open Device

N2Open device is a unit that communicates using N2Open protocol for vendor devices.

Note: For devices that use the System 91 N2 protocol (such as original Metasys hardware), use the "N2 device" class instead.

The form for the N2Open Device is similar to the one for N2 Device. In addition, there is an Online retry delay field. This delay determines how long will the server wait before sending a new "Identify Device Type" message to an off-line device, in case the previous attempt resulted in error, or the device code received did not match the code expected (currently, all device codes are accepted).

Items

The *items* are addressable data elements in the device, made accessible by the OPC server to the OPC clients.

If you have set a hardware type for the device, the device automatically inherits all items defined in that hardware type; such items are not visible in the configuration, but the OPC server presents them in its runtime. The naming and structure of such items is fixed, and is given by the hardware type you have selected. You can still define your own items in the configuration for such device, and they will be

present during runtime in addition to those coming from the hardware type definition. This is typically used if you want to have items with logical names, i.e. names that correspond to your process, instead of following the naming convention used by the physical device.

If you have not set any hardware type for the device (its Hardware Type field is empty), then you definitely need to add some items to it in the configuration, otherwise there won't be any way to access data in such device.

Items of following types can be defined under N2 Device, in any number or combination:

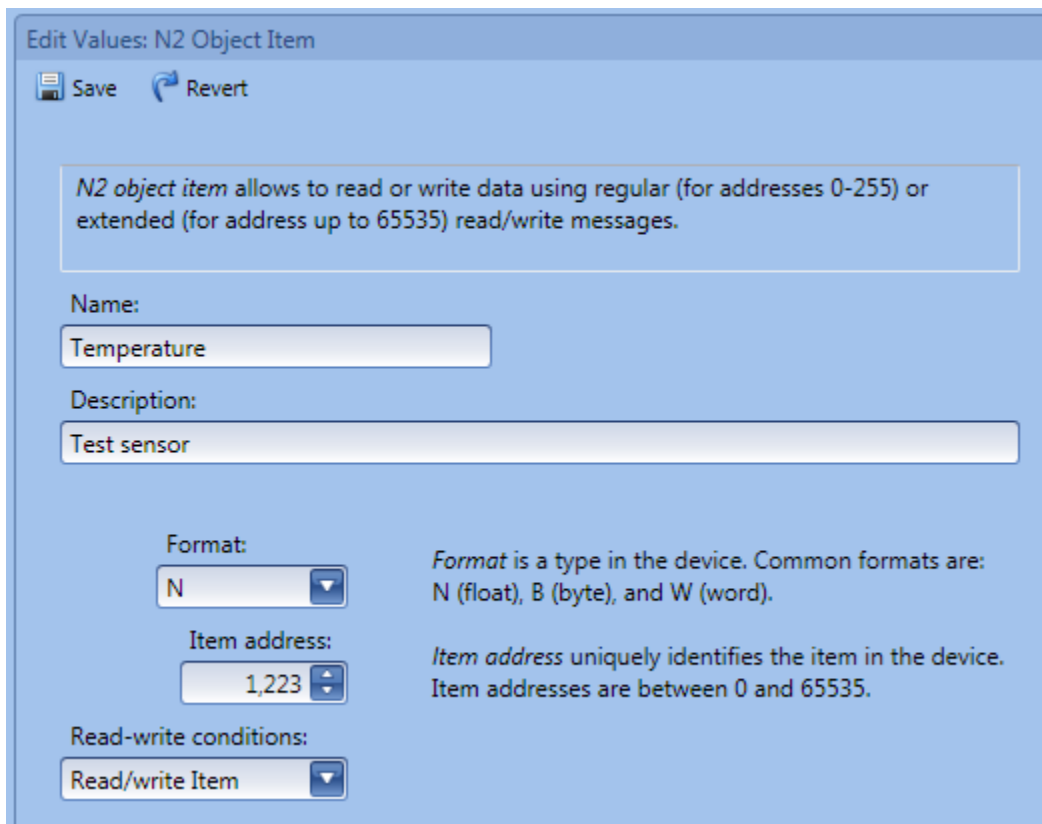
- N2 Object Item
- N2 Functional Module Item (in Hardware Type definition only)

Items of following types can be defined under N2Open Device, in any number or combination:

- N2Open Field Item
- N2Open Command Item (in Hardware Type definition only)

N2 Object Item

N2 object item allows the OPC client to read or write data using regular (for addresses 0-255) or extended (for addresses up to 65535) read/write messages.



Edit Values: N2 Object Item

Save Revert

N2 object item allows to read or write data using regular (for addresses 0-255) or extended (for address up to 65535) read/write messages.

Name:
Temperature

Description:
Test sensor

Format:
N

Item address:
1,223

Read-write conditions:
Read/write Item

Format is a type in the device. Common formats are: N (float), B (byte), and W (word).

Item address uniquely identifies the item in the device. Item addresses are between 0 and 65535.

N2 object items residing in the external hardware type definitions can also be defined as bit-coded fields, i.e. only certain bits of the actual data form the OPC item, and typically there are multiple bit-coded fields defined on a single N2 item. When such items are written into, the OPC server uses a read-modify-write sequence to achieve the desired effect. In this version, you cannot define such bit-coded fields on N2 object items from the configuration application.

N2 Functional Module Item

N2 functional module item allows to read or write data using “Read/Write Functional Modules Blocks Format” message. Different structures of messages exist for Real Time Clock Module, Daylight Savings Module, Exception Days Module, and so on.

In this version, the N2 Functional Module Item can only be specified as part of hardware type definition, and is part of hardware type definitions supplied with the server. You cannot add your own N2 functional module items using the configuration application.

N2Open Field Item

N2Open field item allows the OPC client to read or write data using read/write field messages.

Edit Values: N2Open Field Item

N2Open field item allows to read or write data using read/write field messages..

Name:

Description:

Object address:

Object address is between 1 and 256. Note: Object number (in protocol) = Object address - 1.

Region:

The data in the device are subdivided into regions (0-15).

Attribute number:

Each virtual object may have multiple attributes. Attribute number is between 1 and 255 (0 is reserved).

Type:

Readable Writeable

N2Open Command Item

N2Open command item allows the OPC client to issue special requests in N2Open, such as for overriding the object's primary value, or releasing the override.

The procedure for working with command items (from OPC clients) differs from regular simple reading/writing. For more information, refer to "Commands in N2Open".

In this version, the N2Open Command Item can only be specified as part of hardware type definition, and is part of hardware type definitions supplied with the server. You cannot add your own N2Open command items using the configuration application.

OPC Address Space

The data provided by the OPC server are organized into a hierarchical structure, and exposed in a form of OPC address space. The address space consists of:

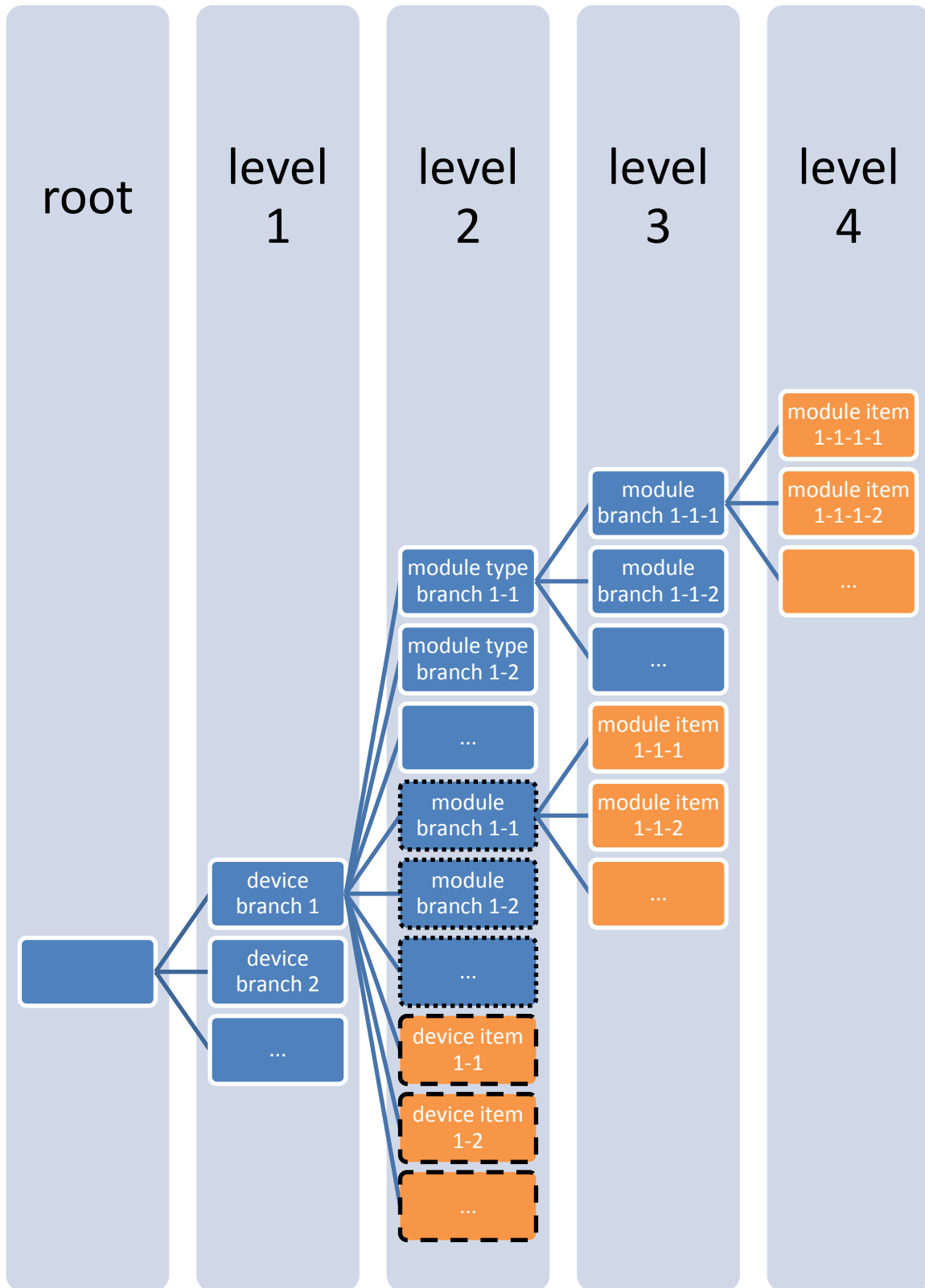
- **Branches.** They only serve an organizational purpose. Branches can contain leaves, or other branches.
- **Leaves.** Those are the terminal nodes that can be used to read, write or subscribe to data. They are sometimes called items, too.
- **Properties.** They are attributes of leaves that expose additional information related to items.

Certain characteristics of the address space are given by the OPC server itself, while the address space remains highly configurable by the user. For example, this OPC server always exposes devices on the first level under the address space root, and this cannot be changed by configuration. However, the actual number of devices and their names are fully configurable.

In order to reduce the amount of nodes, the OPC server has the ability to hide some nodes from OPC clients when they are browsing the address space. When the OPC client knows the precise Item ID, though, the invisible (hidden) items can still be accessed. It is also possible certain leaf can be equivalent in behavior to other leaf in the address space, and that the Item ID provided by the OPC server for them be the same.

This OPC server uses the techniques described above to allow comfortable browsing of the N2 and N2Open address, while keeping the Item IDs relatively short and in line with conventions used in the Metasys system. For example, in order to allow easy navigation for the user, the OPC server may offer a separate branch for all AI (Analog Input) modules, and then list branches for the individual modules AI1, AI2, etc., under the AI branch. When a particular item is selected from one of the AI modules (e.g. AI1), its Item ID does not, however, contain a separate “level” for AI modules; it simply refers directly to the specific module, AI1. This is possible because the branch for AI1 exists as well, but is normally hidden.

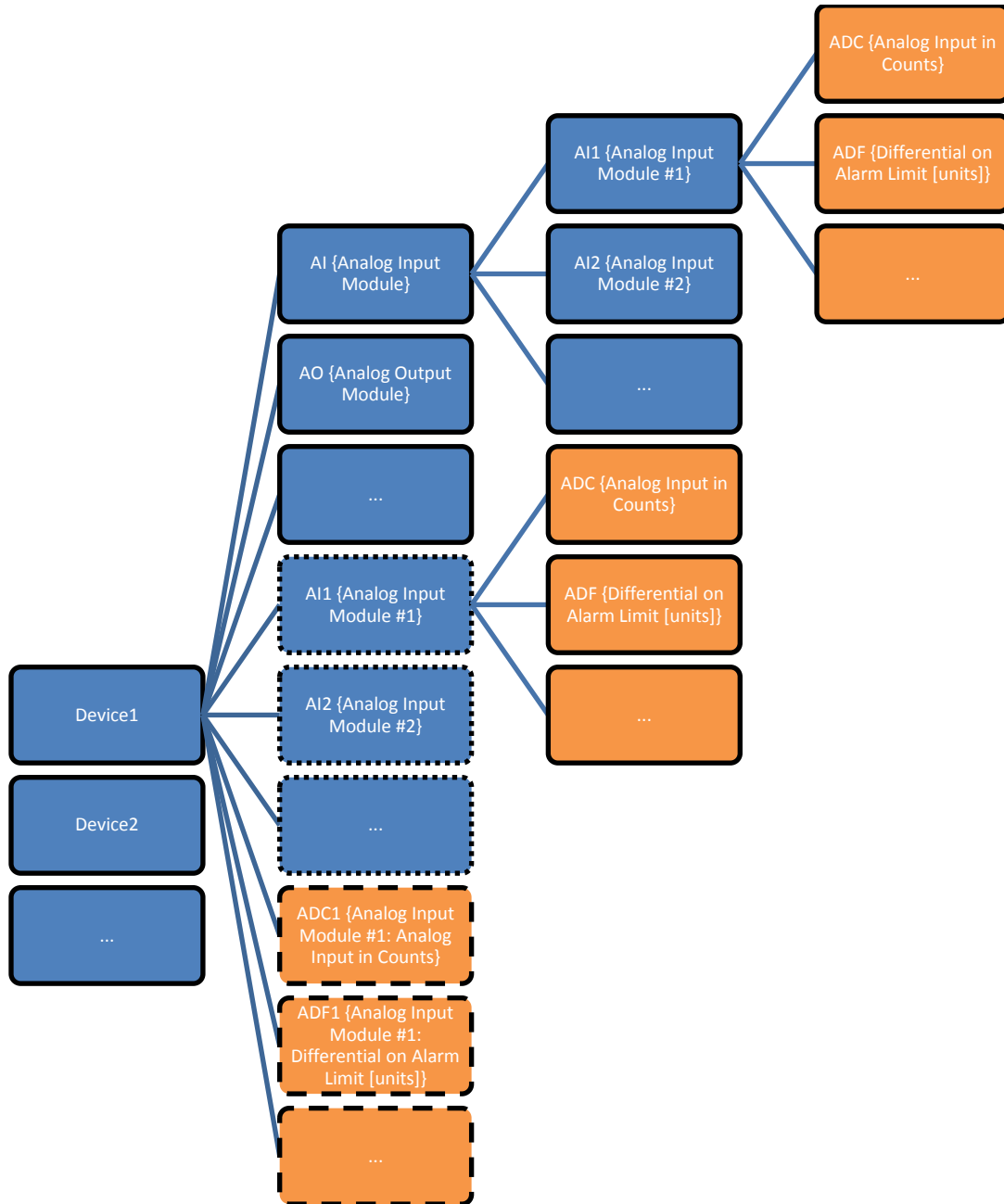
The picture below shows the general structure of address space branches and leaves exposed by this OPC server. Boxes with solid framing are always visible (provided that their parent is visible). Boxes with dotted or dashed framing may be hidden, depending on server settings.



Following convention is used on the figures:

- The address space branches are shown with **blue background**.
- The address space leaves (items) are shown with **orange background**.

The picture below shows a concrete example of OPC address space, with multiple devices. Device1 is the DX-9100 hardware.



It is interesting to note that logically the same items appear three times in the above picture. Also note that under the device level, the item uses a shortened form of its name (ADC1 instead of AI1.ADC).

Names and Data IDs

The node (branch or leaf) *name* is the string that is presented to the OPC client when it is browsing the individual levels of address space hierarchy in the OPC server. It is therefore also what the user will typically see when interactively browsing the address space from a capable OPC client.

The *data ID* is the actual string that will be used as part of the OPC Item ID. In OPC, the data ID may be the same as the name, but does not have to.

This OPC server exposes names as data IDs enhanced by *hints*. The hints are additional pieces of information that describe the purpose of the node. The hint is appended to the data ID, separated by a space character, and enclosed in curly braces. For example, the name may be "AI1", and the corresponding data ID is "AI1 {Analog Input Module #1}". You will see the data ID (with the hint) when browsing, but the hint will not be present in the fully qualified OPC Item ID.

The hint text comes from node description, which is provided either in the hardware type definition, or in the server configuration.

Device Branches

The OPC server creates a branch directly under the root for each device you configure (level 1). The name of the branch is the name you have given to the device in the configuration.

Device Items

Device items are items that appear directly under the device. They can be either explicitly specified, one by one, in the configuration file, or they can be specified by the hardware type definition for the particular device.

Some device items simply provide an alternative access mean to items in specific modules. For example, an ADC1 device item in DX-9100 hardware is the same as AI1.ADC.

Module Type Branches

Module type branches are created by the OPC server to represent modules of the same type (note that module types often mean the same as object types; see explanation in Modules and Module Types).

For example, there are 7 basic module types in a VND hardware, and the OPC server has 7 branches, one for each (ADF, ADI, AI, AO, BD, BI, BO), allowing easy navigation. There are, however, 256 objects (modules) of each object type. Without module type branches, the OPC server would have to present $7 \times 256 = 1,792$ module branches under the device, which would make the navigation quite difficult. The module branches under the device level still exist, but they are hidden by default.

Module Branches

Module branches represent individual modules in the device (note that modules often mean the same as objects; see explanation in Modules and Module Types).

Module branches exist either under their corresponding module type branch (for example, AI1, AI2, etc. module branches under the AI module type branch), or directly under the device branch (such module branches are hidden by default). Note that there is an exception if a module exists only in a single, unnumbered instance of its type (such as the GM module in DX-9100 hardware); in such case, the module branch under the device level is always visible, and there is no module type branch for it.

Module Items

Module items represent the data available in certain module (note that module often means the same as object; see explanation in Modules and Module Types). The module items are always located under a module branch. The set of module items is given by a module type. For example, a module of AI type in DX-9100 hardware always has the same set of module items, such as ADC, ADF, etc.

Properties

Every item defined by an OPC server has specific attributes, or properties, that describe that server item in more detail. These properties include the current Value, Quality and TimeStamp for the server item. Exposure of the server item properties to a client is intended to provide a client with more information on a specific item, and is not intended to provide efficient access to large amounts of data. Rather, you should use the read function to read data from a large number of server items.

Each property is identified by a Property ID, which is an integer value. The OPC Data Access Specification defines some standard and recommended properties; there are also vendor-specific properties.

The table below lists all properties that this OPC server exposes to clients.

Property ID	Name	Data Type	Description	Notes
1	DataType	VT_I2	Item Canonical Data Type	
2	Value	VT_EMPTY	Item Value	
3	Quality	VT_I2	Item Quality	
4	Timestamp	VT_DATE	Item Timestamp	
5	AccessRights	VT_I4	Item Access Rights	
6	ScanRate	VT_R4	Server Scan Rate	
102	HighEU	VT_R8	High EU	
103	LowEU	VT_R8	Low EU	
15000	ItemID	VT_BSTR	Item ID	
15001	Name	VT_BSTR	Item Name	
20000	ErrorCode	VT_I4	Error Code	See Error Codes
20001	ErrorInfo	VT_I4	Error Info	See Error Codes

OPC Operations

The OPC Server attempts to satisfy requests from OPC clients connected to it. The operations that actually affect the target device are OPC reads, writes, and subscriptions.

For OPC reads and writes from/to the device (not from the cache), the OPC server performs the communication necessary to satisfy the OPC request on per-call basis (i.e. OPC reads and writes are not being merged together). For OPC subscriptions, the OPC server determines the fastest requested update rate for each item, and only collects data at this rate (i.e. if multiple subscriptions, and possibly even with differing update rates, the OPC server will collect the data in an optimized manner).

It should be noted that the bandwidth of communication channels between the OPC server and the target device(s) is limited, and the OPC server may not be able to satisfy all requests in a timely manner. If this happens, the rate of periodic data collection (for OPC subscriptions) will slow down below the level requested by OPC clients, and OPC device reads/writes may start returning timeout errors. You should always consider the channel bandwidth, and carefully plan the data collection and one-shot request rates so that channel congestion does not occur or is kept at tolerable levels.

Reading/writing BCD Data in N2

Some N2 devices have functional module blocks that contain items in BCD format. For example, in the DX-9100 controller, such items exist under RTC (Real Time Clock), DS (Daylight Savings), ED (Exception Days) and TS (Time Schedule) modules.

The OPC server exposes 2-byte BCDs as items with VT_I1 data type, and 4-byte BCDs as items with VT_I2 data type. The OPC server also performs the necessary conversions to/from BCD. Obviously, the regular value range for 2-byte BCDs is 0-99, and the regular value range for 4-byte BCDs is 0-9999.

Certain BCD items allow a special value for “not defined” (0FFH for 2-byte BCDs). Such value is represented as -1 (negative one) in OPC. For example, the ED (Exception Days) module requires that these values are set for the start date of the first not programmed period.

Commands in N2Open

In N2Open, analog and binary inputs, analog and binary outputs, and internal parameters can be *overridden* by a special command. The override command is used to send an override value to the object, and the value is to be used in place of its current state. The override value becomes the object's current value.

A previously overridden value can be *released* by a different special command. Once released, the local value is to be used.

On each object that supports this behavior, the OPC Server exposes a pair of special OPC items:

- An 'override' item (e.g. "Device2.AO1.override"). Writing to this item causes the override command be issued, with the override value being the value written to the item. Reading from this item always returns an empty value.
- A 'release' item (e.g. "Device2.AO1.release"). Writing anything into this item causes the override release request be issued. Reading from this item always returns an empty value.

Configuration

The OPC server needs to be configured in order to work properly. The server's runtime part loads the configuration upon startup. The configuration of the server is stored in an XML file with specific schema. For this server, the default configuration file extension is .JCN2.

You may have multiple configuration files, and work with them (also using the configuration tools provided) as you wish. Only one configuration file on the computer can, however, be marked as *active configuration* at any single point. This is the one that the OPC server will load, and information about it (a file path) is stored in the registry.

Be careful not to confuse the configuration you are editing with the active configuration – they may be the same, but not necessarily. The configuration application has commands that allow you to select the active configuration, or view its current setting.

Overview

The configuration application allows you to work with one configuration file (document) at a time.

The user opens an existing document or creates a new one, examines its contents and makes changes as needed, and then saves the document back to the original file, or to a different file.

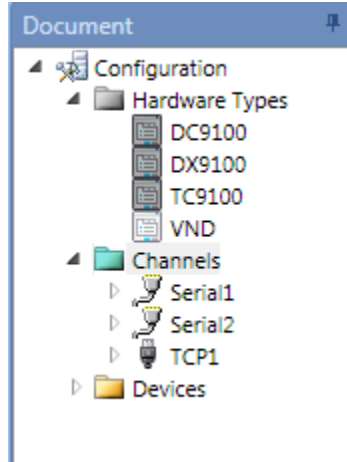
In addition, the configuration application offers some commands for integration with the server runtime. For example, it is possible to re-start the server with newly made changes to the configuration, with a single click of a button.

Panels

The configuration documents in its entirety, and its parts, are displayed in various panels. You can customize the layout of these panels (by dragging their title bars, and moving the splitters) according to your needs. Some panels can be also be docked, or closed.

Document Panel

This panel displays the full structure of the configuration in a tree view. You can expand or collapse the nodes as you wish. Selecting a node displays related information in Node List panel.

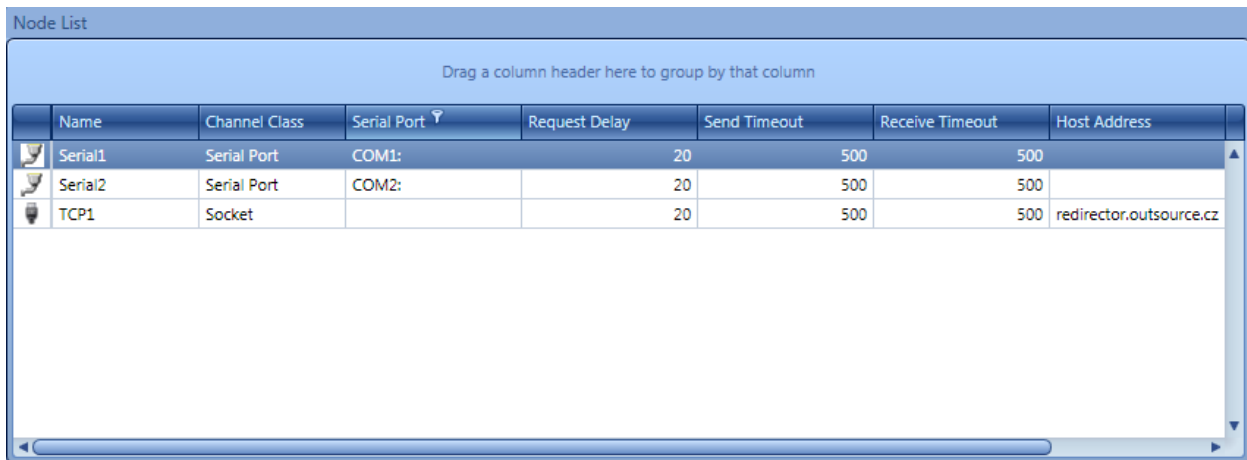


Right-clicking on a node displays a context menu with commands related to that node. It is quite typical to invoke this context menu in order to insert new sub-nodes.

Node List Panel

The Node List panel displays objects in a grid, one object per row. The objects displayed are either sub-nodes of the node selected in the Document panel, or (if the selected cannot have any sub-nodes) the selected node itself is displayed.

Selecting a row in the grid causes the detailed form for that row to be displayed in the Edit Values panel.



The Node List panel displays a table with the following data:

Name	Channel Class	Serial Port	Request Delay	Send Timeout	Receive Timeout	Host Address
Serial1	Serial Port	COM1:	20	500	500	
Serial2	Serial Port	COM2:	20	500	500	
TCP1	Socket		20	500	500	redirector.outsource.cz

The grid offers various functionalities for easy navigation and viewing, such as:

- Object icons.
- Customizable column widths.
- Automated best fit column widths (for selected column, or all columns).
- Sorting by a column (ascending or descending).
- Hiding/showing columns (Column Chooser).

- Filtering by column values.
- Grouping by column values.

Right-clicking on a row displays a context menu with commands related to that row. It is quite typical to invoke this context menu in order to e.g. delete existing rows.

By clicking any of the column headers, you can sort the grid by data in that column, either in ascending or descending order.

If you hover the mouse pointer over any of the column header, a little filter icon is appears in the header. By clicking this icon, you are offered a list of unique values in that column, and you can filter the grid by selecting a value of your interest.

Column can be resized by dragging the vertical dividers in the header area. Double-clicking the divider to the right side of the column causes the column to resize accordingly to the data it contains, choosing the best fit.

It is possible to group the rows in the grid by dragging a column header to the marked area above the column headers. You can group by one or more columns. Groups are collapsible and expandable.

Right-clicking on a column header displays a context menu with commands related to particular column.

Edit Values Panel

The Edit Values panel contains a form that allows you to view or modify the data for a row that is selected in the Node List panel (in the grid). The form displayed in this panel depends on the type of configuration object selected.

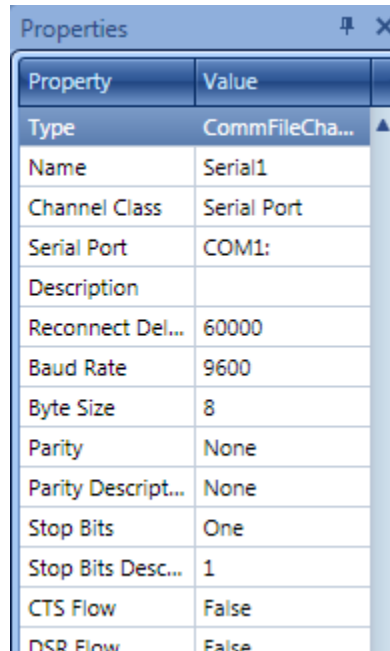
The form may have multiple tabs, or expandable areas. The data entered is validated on several levels (each control as being edited, control values and their consistency when the data is being saved, and the data is also checked against the XML schema of the file). A notification icon or an error box appears when invalid data is entered.



Press the Save button to store the changes you have made in the controls of the Edit Values panel back into the document. Press the Revert button to revert the changes you have made in the controls, and thus cancel any modification to the document.

Properties Panel

The Properties panel displays all properties of the object that is currently selected in the Node List panel.



Property	Value
Type	CommFileCha...
Name	Serial1
Channel Class	Serial Port
Serial Port	COM1:
Description	
Reconnect Del...	60000
Baud Rate	9600
Byte Size	8
Parity	None
Parity Descript...	None
Stop Bits	One
Stop Bits Desc...	1
CTS Flow	False
DSR Flow	False

Command Groups

This chapter describes the various commands available in the configuration tool. Most commands are available as menu items, and we group them by the title on the main menu bar. Some commands have their associated icons or other entries on toolbar(s) as well.

File Commands

Commands in the File group offer standard set of New, Open, Close, Save, Save As, and Exit operations, together with a list of most recently used files.

In addition, the Mark Active command marks the currently opened document as the configuration that the server runtime should use. The runtime will load this file next file it is started.

Edit Commands

Commands in the Edit group offer standard set of Copy, Cut and Paste operations. Note that pasting is highly context-sensitive, i.e. you can only Paste objects of types that are allowed under currently selected node.

The Edit group also has commands that work on the node or row level, such as those that come from the context menu invoked on a node in a Document panel, or on a row in a Node List panel:

The New Node submenu offers node types that can be added under the current node.

The Delete Node command deletes the current node, and everything beneath it.

The Multiple Node command allows you to create nodes that are copies of the current node, with modifications such as sequential numbering.

The Select All Rows, Unselect All Rows and Invert Selection commands allow you to select or unselect large numbers of rows in the grid by a single operation.

The Delete Rows command allows you to delete all rows that are currently selected in the grid.

View Commands

The commands in the View group influence the user interface of the configuration application. They have no effect on the contents of the file being edited.

With the commands in this group, you can:

- Show or hide various toolbars.
- Show or hide certain panels.
- Invoke a dialog that allows you to customize the toolbars, and select additional options for the application look.

Tools Commands

The commands in the Tools group allow additional functionality that is not directly related to the usual file contents editing tasks.

The Server Settings command invokes a separate utility that configures behavior of the server runtime part. See N2 OPC Settings Utility chapter for details.

If the OPC service is registered as a service (see COM Registration and Server Types), you can use Start Service, Stop Service and Restart Service commands to control the server.

The Execute Configuration command (also only available when the OPC server is registered as a service) allows you to quickly instruct the server to take over whatever configuration you currently have in the application. This command effectively performs following steps:

1. Save the configuration (allowing you to give it a name if it is a new file).
2. Mark the configuration as the active configuration, for the runtime part.
3. Start the service, if it is stopped; or restart (stop and start) the service, if it is running.

The Preferences command invokes a dialog that contains various settings that influence the behavior of the application. The settings include:

- Open last loaded file at startup.
- Expand all groups in node list grid.

Help Commands

The Help group offers the About command, which displays information about the configuration application, including its precise version and build number.

Operations

This chapter describes how the OPC server is operated, after being properly deployed and configured as needed for the particular purpose.

OPC Client Connections

Different OPC clients have different procedures for specifying the connection to OPC server. Most clients will allow you to browse the computer for available servers, but some may require the connection data be entered manually.

Following table contains information that is needed by various tools to instantiate the OPC server:

Class Name	CLSID	ProgID	Version Independent ProgID
JCN2Server	74E79D99-D296-4cc7-B718-AFC7EB31FB96	OPCLabs.JCN2OPCServer.1.0	OPCLabs.JCN2OPCServer

Server Control

When the OPC server is registered as Local Server, the COM/DCOM infrastructure starts it automatically when any COM object from it is required by the clients, if it's not running. When registered as Local Server, the OPC server terminates itself after certain period of inactivity, if it is no longer serving any COM objects.

When the OPC server is registered as Service, you can control it as other services in the system. Note that unless the startup type is set to Disabled, the COM/DCOM infrastructure also starts the OPC server for you if it's not running and any COM object from it is required by the clients (this happens even when the startup type is Manual).

To start the N2 OPC Server from the configuration application (when registered as Service), invoke the Tools -> Start Service command.

To manually start the N2 OPC server from the command line (when registered as Service):

- Open the Command Prompt window.
- Type the following command:
`NET START JCN2OPCServer10`

To stop the N2 OPC Server from the configuration application (when registered as Service), invoke the Tools -> Stop Service command.

To stop the N2 OPC server manually from command line (when registered as Service):

- Open the Command Prompt window.
- Type the following command:

```
NET STOP JCN2OPCServer10
```

Troubleshooting

If the OPC server encounters an error while trying to access a data in the device, it reports the problem using the means of OPC errors and qualities. In order to allow for more fine-grained diagnostics, the precise cause of the problem is exposed by means of two server-specific OPC properties that exist on every OPC item that corresponds to data in the device. These properties are called Error Code and Error Info. For more information about these properties, see Error Codes further below.

In some cases, mainly for issues of wider scope, the server reports errors (and other relevant occurrences) to an event log (see Event Logging).

Error Codes

The error codes listed in the table below are reported in the Error Code property (Property ID 20000, data type VT_I4) available on each item that corresponds to data in the device. Some error codes are accompanied by an additional value in Error Info property (Property ID 20001, data type VT_I4); the precise meaning of this error information depends on the error code, and is also explained in the table below.

Error Code	Source	Description	Error Info
-2	N2 or N2Open runtime	Protocol warning	0
-1	N2 or N2Open runtime	Circuit controller warning	0
0	-	OK (no error or warning)	0
1	N2 runtime	Circuit controller error	0
2	N2 runtime	Protocol error	0
3	N2 runtime	Station missing	0
4	N2 runtime	Station disabled	0
5	N2 runtime	Station suspended	0
6	N2 runtime	Cannot convert to N2 float	0
7	N2 runtime	Cannot convert to N2 BCD	0
8	N2 runtime	Unknown functional module type	0
9	N2 runtime	Improper index sequence length	0
10	N2 runtime	Improper index sequence character	0
51	N2Open runtime	Circuit controller error	0
52	N2Open runtime	Protocol error	0
53	N2Open runtime	Station missing	0
54	N2Open runtime	Station disabled	0

55	N2Open runtime	Station suspended	0
56	N2Open runtime	Station offline	0
57	N2Open runtime	No command written	0
58	N2Open runtime	Cannot convert to N2Open data	0
59	N2Open runtime	Cannot convert to command input	0
101	Circuit controller	Circuit error	Circuit error info
102	Circuit controller	Inhibited	0
103	Circuit controller	Connect timeout	0
201	Socket transceiver	Error in startup	Winsock error code
202	Socket transceiver	Error getting host by name	Winsock error code
203	Socket transceiver	Incorrect address length in host entry	0
204	Socket transceiver	Error creating socket	Winsock error code
205	Socket transceiver	Error connecting	Winsock error code
206	Socket transceiver	Error creating event	Winsock error code
207	Socket transceiver	Error sending	Winsock error code
208	Socket transceiver	Error receiving	Winsock error code
221	File transceiver	Error creating event	Win32 error code
222	File transceiver	Error creating file	Win32 error code
223	File transceiver	Error writing to file	Win32 error code
224	File transceiver	Error reading from file	Win32 error code
301	N2 or N2Open protocol	Transient communication error	0
302	N2 or N2Open protocol	Permanent communication error	0
303	N2 or N2Open protocol	Odd data size	0
304	N2 or N2Open protocol	Data too short	0
305	N2 protocol	BCC mismatch	0
396	Optomux communicator	Response data too long	0
397	Optomux communicator	Data not hexadecimal	0
398	Optomux communicator	Wildcard checksum not allowed	0
399	Optomux communicator	Error response returned	Optomux error code
401	Optomux communicator	Send fault	0
402	Optomux communicator	Send timeout	0
403	Optomux communicator	Receive fault	0
404	Optomux communicator	Receive timeout	0
405	Optomux communicator	Error code syntax	0
406	Optomux communicator	Empty message	0
407	Optomux communicator	Invalid message string	0
408	Optomux communicator	Message syntax	0
409	Optomux communicator	Message not terminated	0
410	Optomux communicator	Characters after terminator	0
411	Optomux communicator	Invalid response outcome	0
412	Optomux communicator	Response body too short	0
413	Optomux communicator	Checksum syntax	0

414	Optomux communicator	Incorrect checksum	0
415	Optomux communicator	Invalid start of message	0
416	Optomux communicator	Address syntax	0

For error codes that originate in the devices, see Appendix A. Optomux, N2 and N2Open Error Codes.

Advanced Topics

OPC Specifications

The N2 OPC server directly supports following OPC specifications:

- all OPC DA (Data Access) 2.0x Specifications (Released)
- all OPC DA (Data Access) 3.0x Specifications (Released)
- all OPC Common 1.0x Specifications (Released)

The N2 OPC server supports following OPC specifications indirectly:

- OPC UA (Universal Architecture) 1.00 Specifications for Data Access
- OPC UA (Universal Architecture) 1.01 Specifications for Data Access

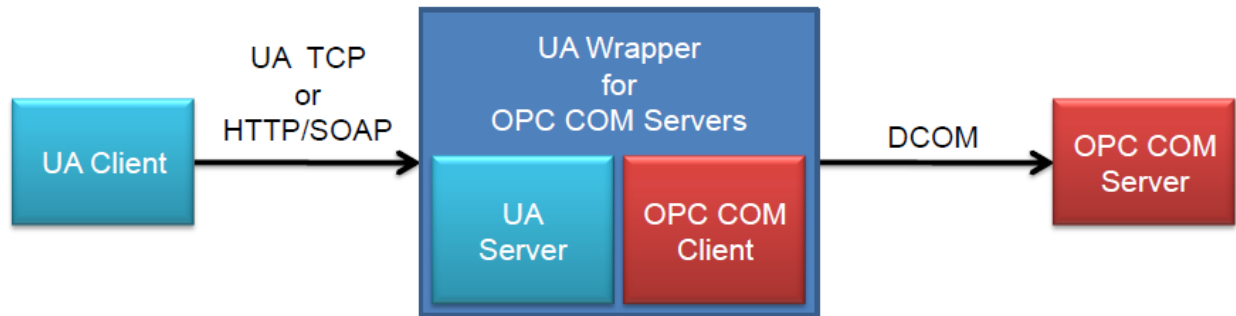
OPC-UA (Universal Architecture)

The Unified Architecture (UA) is the next generation OPC standard that provides a cohesive, secure and reliable cross platform framework for access to real time and historical data and events.

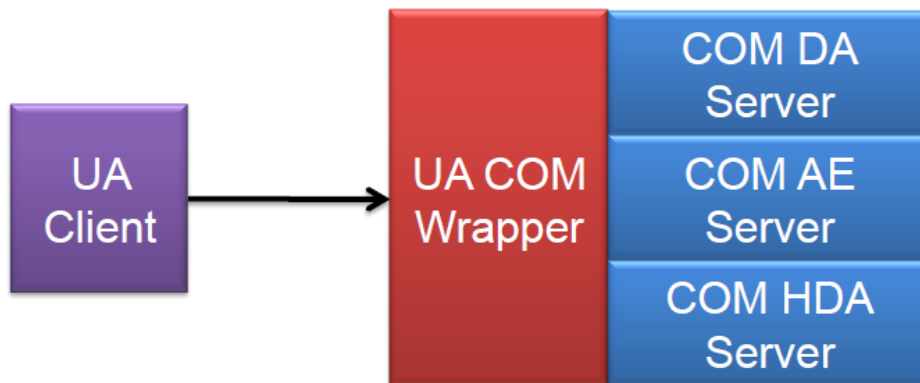
OPC Server for Johnson Controls N2 is not a native OPC-UA server, but it can be made to act as such using so-called UA COM Wrapper that is shipped with the product as part of OPC UA COM Interop Components. Using the wrapper, the server becomes a fully compliant OPC-UA server, and you can connect OPC-UA clients to it.

The OPC UA COM Interop Components from OPC Foundation make it possible for existing COM-based applications to communicate with UA applications. OPC Labs is using them to add UA support to existing products.

The UA Wrapper (or UA COM Wrapper) is designed to allow a UA client to talk to a COM server. It consists of a UA server, which can be configured to wrap one or more COM servers. Whenever a UA client connects to this UA server, all the calls are routed over using DCOM to the COM server. The wrapper itself is simply a thin layer that translates the UA calls into calls on a COM server, so all the information is coming dynamically from that COM server at the end of the line.



The UA COM Wrapper is a UA Server that can wrap multiple COM servers with the same or different CLSIDs. The COM server connection information is contained in the wrapper configuration file. Note that the UA COM Wrapper has to be restarted after changes to its configuration, in order for it to pick up the new configuration data.



Note: The setup program has the “OPC UA COM Interop Components” option turned off by default, because it has other dependencies and effects on the system that can complicate the typical setup. If you want to connect to OPC-UA servers, make sure that you enable “OPC UA COM Interop Components” in the installation first.

Configuring the Server for OPC-UA

Perform following steps in order to configure the server for OPC-UA, wrap it as OPC-UA server, and make it available to OPC-UA clients:

1. Start UA Configuration Tool. It can be found in your Start menu, under OPC Foundation → UA SDK 1.01 → UA Configuration Tool.
2. In the UA Configuration Tool, select the “Manage COM Interop” tab, and press the “Wrap COM Servers” button.
3. In the “Managed Wrapped COM Servers” dialog, press the “Add” button.
4. In the “Select a COM Server to Expose via UA” dialog, select the line with parameters listed below, and press “OK” button.

- Prog ID: OPCLabs.JCN2OPCServer
 - Server Name: OPCV Labs Johnson Controls N2 OPC Server 1.0
 - Specification: Data Access 3.00
5. In the “Managed Wrapped COM Servers” dialog, select the line with Browse Name “OPCLabs.JCN2OPCServer”, and press “Edit” button.
 6. In the “Edit Wrapped COM Server” dialog, enter “.” (period) into the “Seperator Chars” field, and press OK.
 7. In the “Managed Wrapped COM Servers” dialog, press the “Close” button.
 8. Exit from the UA Configuration Tool.
 9. Restart the UA COM Server Wrapper. You can use either of the following procedures.
 - a) Use Services in Computer Management console, locate “UA COM Server Wrapper”, and press “Restart the service” link.
 - b) In Command Prompt, enter
NET STOP “UA COM Server Wrapper”
followed by
NET START “UA COM Server Wrapper”

OPC Interoperability

The OPC server has been extensively tested for OPC interoperability, with various OPC clients from many vendors, and on different computing environments.

Having tried so many different OPC clients, we have encountered different (though still correct) interpretations of the same OPC specifications, and also certain common (and less common) divergences from the specifications. Wherever possible while still remaining compliant, the OPC server chooses to implement OPC in such a way that is compatible with most OPC clients. This gives the OPC server even wider interoperability scope.

Event Logging

The OPC server is capable of logging various errors and events, using the standard Windows mechanisms, or other means. By default, only the most important events are logged.

You can use the Event Log Options utility (available from Start menu) to influence which events get logged, and also to select logging into a plain text file instead of Windows Even Log. The Event Log Options utility has separate documentation and help file; please refer to it for details on setting the options.

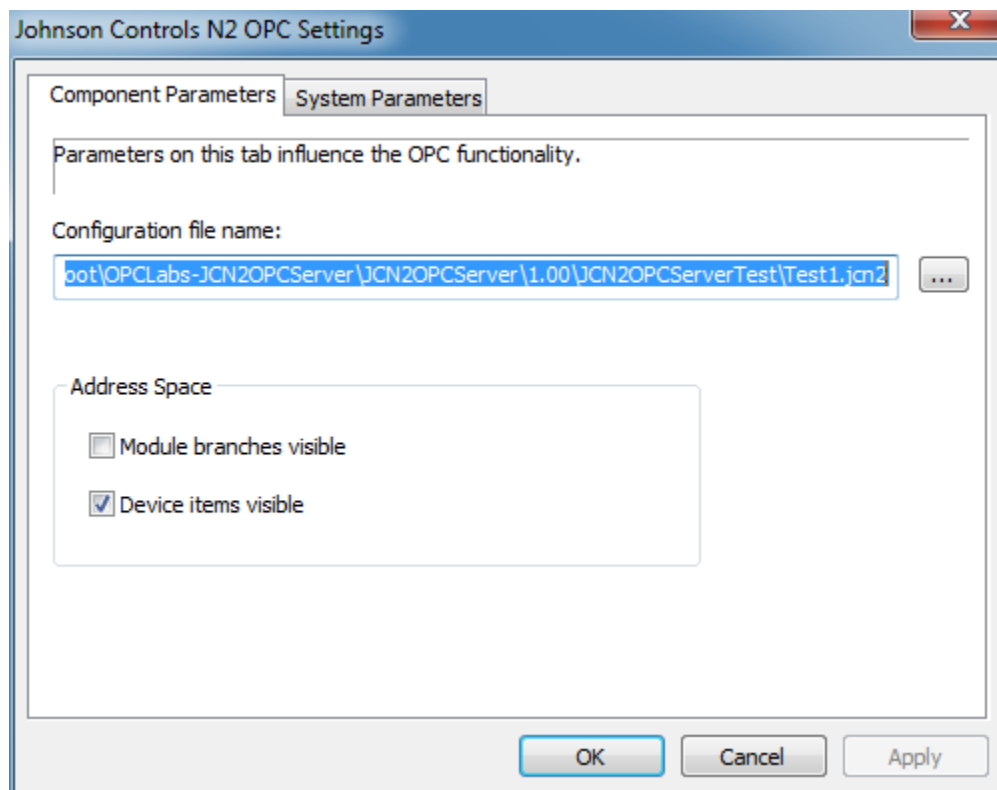
N2 OPC Settings Utility

The OPC server comes with predefined settings that are suitable for most applications. For large-volume operations, or specialized needs, it may be necessary to fine-tune the settings, using the N2 OPC Settings utility. You can invoke the utility from the Start menu, under the application's program group, or by Tools -> Server Settings command from the configuration application.

Be aware that the OPC server only "picks up" the settings at the startup time. You should therefore set the proper options in advance, and start the OPC server afterwards.

Component Parameters Page

This tab contains the main parameters that influence the behavior of OPC server with regard to data collection, and how the data is presented to OPC clients.



Configuration file name

Specifies the path and name of the configuration file that the server will load and execute.

This is the same setting that can be achieved by invoking the File -> Mark Active command from the configuration application, or happens as one of the effects of Tools -> Execute Configuration command.

Address Space Group

This group contains settings that influence the address space of the OPC server, i.e. the hierarchical structure of OPC items available for browsing and connections to OPC clients.

Module branches visible

This setting determines whether module branches (see Module Type Branches

Module type branches are created by the OPC server to represent modules of the same type (note that module types often mean the same as object types; see explanation in Modules and Module Types).

For example, there are 7 basic module types in a VND hardware, and the OPC server has 7 branches, one for each (ADF, ADI, AI, AO, BD, BI, BO), allowing easy navigation. There are, however, 256 objects (modules) of each object type. Without module type branches, the OPC server would have to present $7 \times 256 = 1,792$ module branches under the device, which would make the navigation quite difficult. The module branches under the device level still exist, but they are hidden by default.

Module Branches) on the device level are visible during address space browsing. When the module branches are not visible, the OPC client does not see them while browsing the parent level, however they still exist and are accessible by their data ID if the client specifies it.

Making module branches invisible improves the performance of browsing.

Note that the module branches on the module type level are always visible.

Example: Under DX-9100 device named Device1, a module type branch AI can be seen, for all analog input modules. Further under the AI branch, a module branch AI2 can be seen, for the second analog input module. The same AI2 branch can be seen directly under the device for quicker access, however this branch (directly under Device1) can be made invisible for browsing, reducing the large number of branches on the device level.

Device items visible

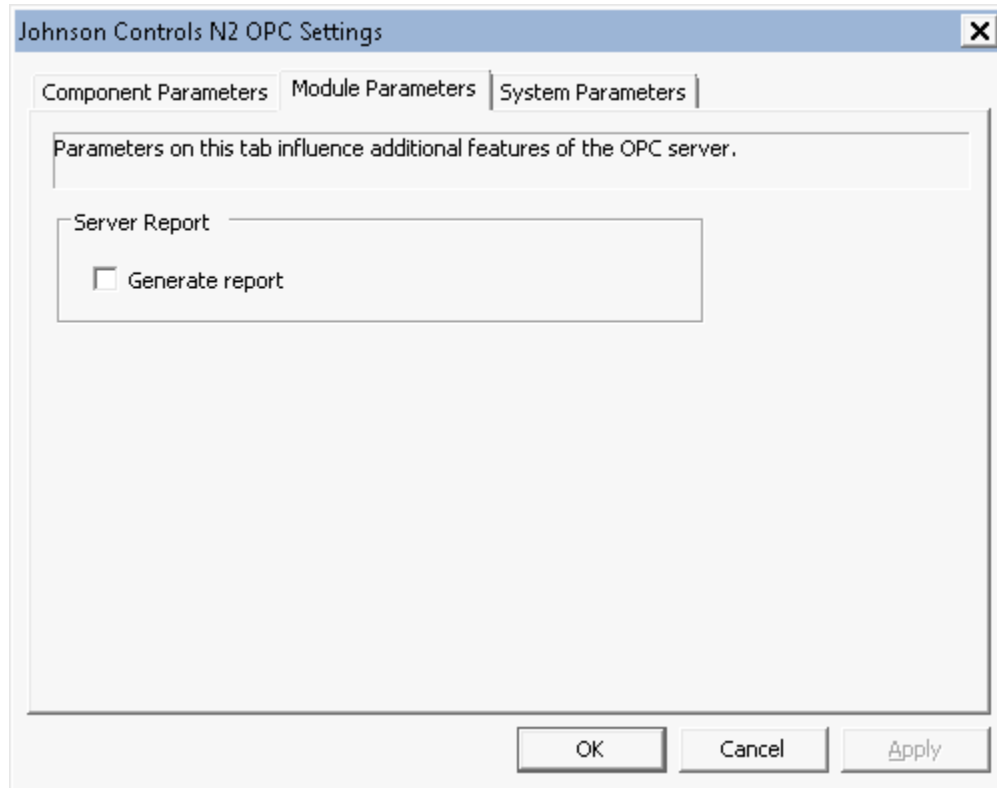
This setting determines whether Device items (see Device Items, on the device level) are visible during address space browsing. When the device items are not visible, the OPC client does not see them while browsing the parent level, however they still exist and are accessible by their data ID if the client specifies it.

Making device items invisible improves the performance of browsing.

Example: For DX-9100 device named Device1, the % value of second analog input is accessible as Device1.AI2.AI%. The same item can be accessed directly under the device as Device1.AI2%, however this device item can be made invisible for browsing, reducing the large amount of items on the device level.

Module Parameters Page

Parameters on this tab influence additional features of the OPC server.



Server Report Group

This group contains settings that affect the generation of server report. This report details any problems found during configuration loading.

If report generation is enabled, the server creates one report file each time it starts. The reports are text files stored in the Reports subfolder under the product installation folder. Report files are named "opcYYYYMMDDhhmmss.txt", where the letters in *italics* are replaced by current date and time (of the moment when the server starts).

Generate report

The server report will be generated if this checkbox is checked.

System Parameters Page

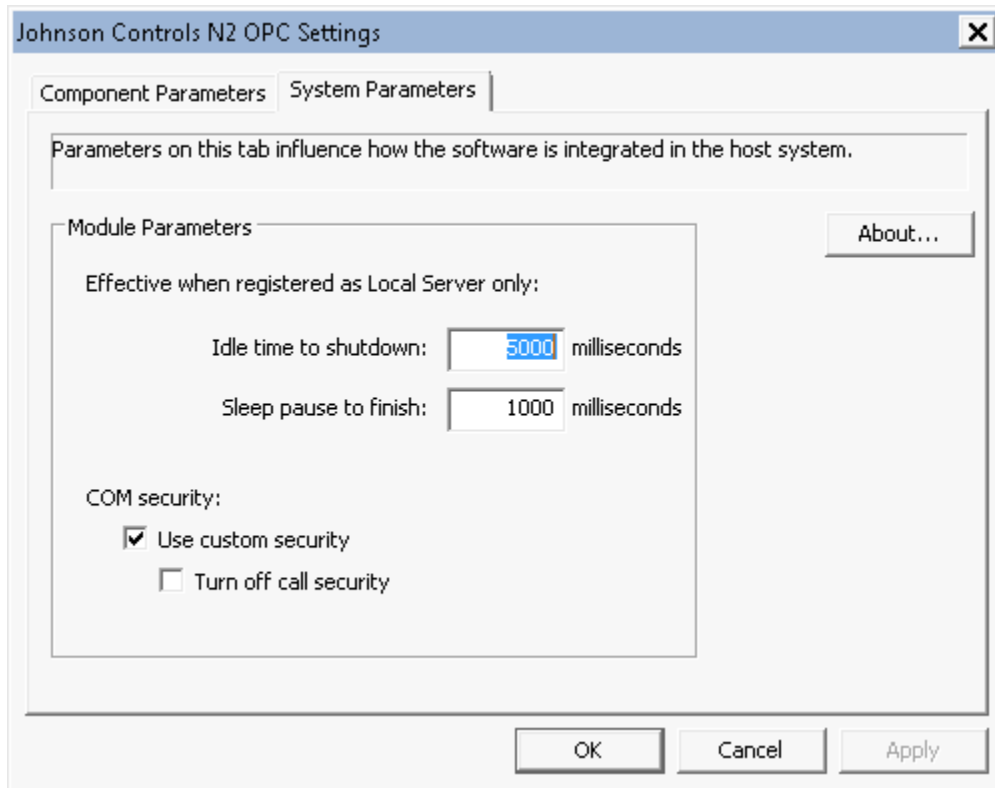
Parameters on this tab influence how the software is integrated in the host system.

About

This button displays a window with the copyright and version information about the utility.

Module Parameters Group

This group contains settings that affect the module (a binary code loaded from .EXE and .DLL and executing) as a whole.



Idle time to shutdown

The N2 OPC local server will quit if there are no objects left to serve and there is no activity for the period specified here.

Sleep pause to finish

A delay given to any remaining threads to finish before the module finally stops.

COM security

Use custom security

When checked, the COM security settings of the application are controlled by the application itself, and are not influenced by the settings specified in DCOM Configuration. Unless specified otherwise, access is allowed to everyone, authentication level is set to Connect and the impersonation level is set to Impersonate.

Turn off call security

Sets the application's authentication level to None.

COM Registration and Server Types

The OPC server can be deployed as following COM server types:

- Local Server, or
- Service.

There are various advantages and disadvantages of the above options, which are discussed in details in Microsoft COM materials. In brief, the Service can be better controlled while running (started, stopped, etc., using the Service Manager). The OPC server can be registered only as one server type at the same time.

By default, the installation program registers the N2 OPC server as Service.

To register the N2 OPC server to run as Local Server (and not as Service):

- Open the Command Prompt window.
- Navigate to the “bin” subdirectory of the N2 OPC server installation folder.
- Type the following command:

```
jcn2os /RegServer
```

To register the N2 OPC server to run as Service (and not as Local Server):

- Open the Command Prompt window.
- Navigate to the “bin” subdirectory of the N2 OPC server installation folder.
- Type the following command:

```
jcn2os /Service
```

Internal Optimizations

The OPC server contains a number of internal optimizations designed to make its operations efficient even with large data volumes.

Main optimization performed with regard to OPC clients is a consolidation of multiple subscriptions for the same item into a periodic read with a single common update rate.

Failure Recovery

The server is provided with auto-recovery algorithms for all types of external failures that can occur during its operations. The specific types of failures, and their handling, are described below.

Protocol Errors

Protocol errors are most error responses returned by the device (except responses that indicate a corrupted request or bad checksum), and failures to properly interpret a message that otherwise conforms to protocol rules (its overall integrity has been verified, but the server still does not understand the message).

A protocol error encountered at any time does not have influence on further operations of the server. The requested operation simply fails immediately (no retries), and an error code corresponding to the specific error is returned. For single-shot OPC reads and writes, this is the “end of story”. For OPC subscriptions, the same operation is repeated periodically as usual (the period depends on the requested update rates).

Transient Communication Errors

A transient communication error may occur as a result of mechanical (e.g. broken wire), electrical (e.g. noise induction) or other unavoidable problem in production. These errors happen in following situations:

- The server fails to send a request or receive a response during certain time (a timeout).
- The server receives a response message that appears corrupted (does not conform to protocol rules for valid messages), or has an incorrect checksum.
- The server receives an error response indicating that the request s arrived corrupted, or with an incorrect checksum.

When a transient communication error is detected, the server retries the operation by sending out the same request again. The number of retry attempts is controlled by the “Retries before suspend” setting, separately for each device. If, even after the configured number of retries, the transient communication error still persists, the device is set to Suspended mode. In the Suspended mode, the server does not physically attempt any operations to the device, and instead returns an error indicating the device status (i.e. Suspended). The Suspended mode is cancelled (the device communication is resumed) after a delay, configured by “Delay before resume” value for the device.

Channel Errors

Channel error occurs when the server encounters a failure that prevents it from opening or communicating through a channel (serial port, or socket). For example, the serial port may be already opened by another process on the machine, the configured IP address may be wrong, or the Ethernet/serial convertor encounters a loss of power.

All such failures cause the channel to become unavailable for certain time, and if an operation on any device connected to this channel is requested, the server returns an error indicating the channel failure. After a “Reconnect delay” (configured for the channel) elapses, the server will attempt to re-open the channel and communicate through it again.

Data Types

The N2 OPC Server converts data to and from OPC according to the tables below. For N2 protocol:

N2 Data Type	VARTYPE In OPC
Floating point number (16 bits JC format)	VT_R4
Unsigned 8 bits	VT_UI1
Unsigned 16 bits	VT_UI2
Unsigned 32 bits	VT_UI4
2-digit BCD	VT_I1
4-digit BCD	VT_I2

For N2Open protocol:

N2Open Data Type	VARTYPE In OPC
Float (32 bits IEEE format)	VT_R4
Integer (16 bits signed)	VT_I2
Integer (32 bits signed)	VT_I4
Byte (8 bits unsigned)	VT_UI1

In all cases, strings are internally represented in Unicode wherever possible.

Multithreading and Synchronization

The OPC server COM object and all its related objects are thread-safe, i.e. the OPC client code does not need to perform any external synchronization to access these objects.

The OPC server is internally multithreaded with respect to various functionalities it handles. Specifically, access to devices residing on separate channels is fully parallelized. This means that the OPC server communicates simultaneously on multiple configured serial ports or TCP sockets, to provide maximum data collection performance.



Additional Resources

We recommend that you check the vendor's Web page for updates, news, related products and other information.



Appendix A. Optomux, N2 and N2Open Error Codes

Slave devices communicating with Optomux protocol (and, consequently, N2 and N2Open protocols) uses error response messages to indicate problems to the master. The error response contains error code (1 byte).

Optomux protocol defines several error codes, and N2 and N2Open protocols add their own error codes. In case of problems indicated by the error response, the server provides the error code to the clients, for troubleshooting purposes.

The tables below contain error codes defined by Optomux, N2 and N2Open protocols, respectively.

Optomux Error Codes

Error Code	Description
00H (0)	Power-Up Clear Expected
01H (1)	Undefined Command.
02H (2)	Checksum Error
03H (3)	Input Buffer Overrun
04H (4)	Non-printable ASCII Character Received
05H (5)	Data Field Error
06H (6)	Communications Link Watchdog Timeout Error
07H (7)	Specified Limits Invalid

N2 Error Codes

Error Code	Description
80H (128)	Data not matching the item or function type
81H (129)	Not existing item or function
82H (130)	Temporarily impossible to access the item
83H (131)	Not programmable item
84H (132)	Table programmed with illegal items
85H (133)	Trend programmed with illegal item
86H (134)	Invalid Functional Module
87H (135)	Exceeding Addressing Range
88H (136)	Undefined Address after gate
89H (137)	No answer from Device after gate
8AH (138)	Password Protection Active

N2Open Error Codes

Error Code	Description
10H (16)	Invalid Data
11H (17)	Invalid command for data type
12H (18)	Command not accepted

