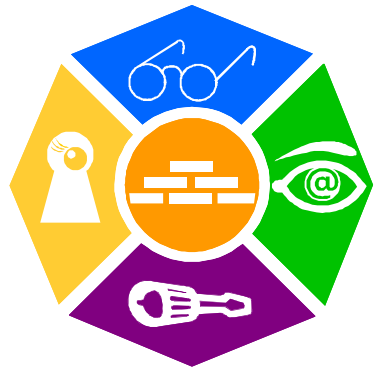
NLOPC-VNI
MANUAL



NEWRON SYSTEM
25-27 Boulevard Victor HUGO
31770 COLOMIERS (France)
T: +33 (0)5 61 15 18 45
F: +33 (0)5 61 15 16 44

SUMMARY

PICTURES

INTRODUCTION

Thank you for choosing NLOPC-VNI software member of NLSuite.

We are happy to help you in your LonWorks integration. All software of NLSuite are often updated to correct bugs and improve performances. Please check version on Web site www.newron-system.com.

NL220

This is a LonWorks LNS Manager tool.

NLFacilities

This is a graphical tool for managing your living spaces.

NLOPC-MIP

This is a very fast OPC server with embedded tool to tune your Scada.

NLOPC-VNI

It is a LNS OPC server. It can manage directly iLon interfaces.

NLUtil

This is a window node utility. It is used before installation for checking channel and other LonWorks products.

INSTALLATION OF PROGRAMS

This section explains how to install the NLOPC-VNI program

Configuration requirements

The table below shows the minimum configuration and the recommended configuration for the installation and correct functioning of the program.

Equipment	Minimum	Recommended
Operating system	Windows 95, 98, Me NT, 2000, Xp	Windows NT, 2000, Xp
Computer	Pentium III 350 Mhz, 800 x 600 screen	Pentium III 750 Mhz, 1024 x 768 screen
Memory	64 M octets	128 M octets
Hard disk	10 Mo	20 M octets
CD ROM	Required for installation	Required for installation
Software	- LNS 3.0 or greater	- LNS 3.0 or greater
Interface network	Type NSI or VNI card	Type NSI or VNI card

Table 1 The equipment

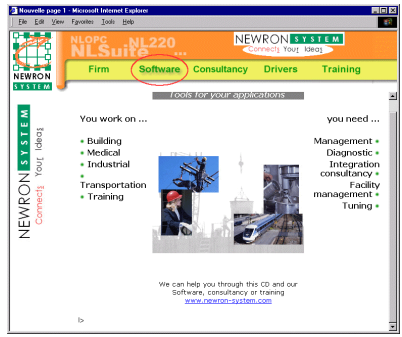
Installation

A setup program will guide you through the installation procedure and will ask you for any information necessary.

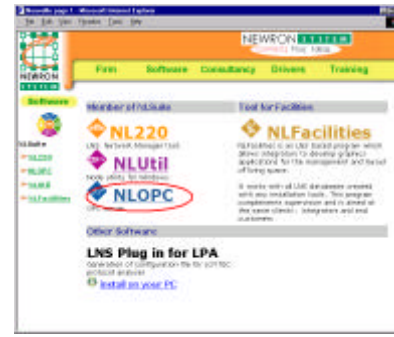
Installation of program

1. Insert the CDROM in the CD reader
2. If no window appears on the screen open: D:\index.htm
3. Picture 1 will appear on the screen.
4. Select **Software** on main Menu
5. Picture 2 will appear on the screen.
6. Select **NLOPC** on center of screen.
7. Picture 3 will appear on the screen.
8. Select **Click to install NLOPC-VNI on your PC**, Picture 4 will appear on the screen.

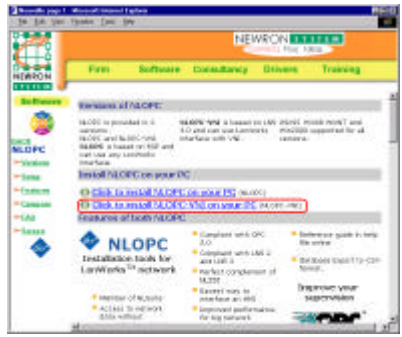
9. Select **run this program fro its current location** and click on **OK** button.



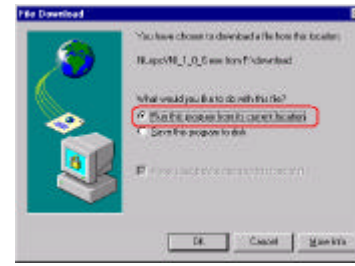
Picture 1 Install Step 1



Picture 2 Install Step 2

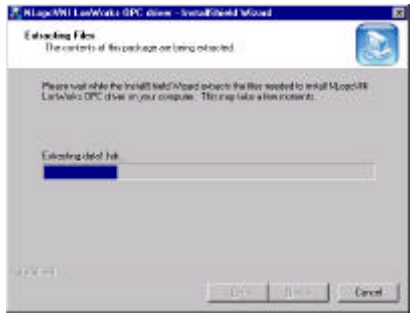


Picture 3 Install Step 3



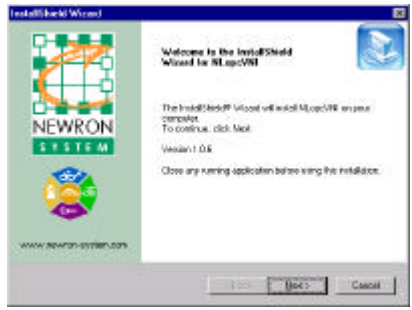
Picture 4 Install Step 4

The installation program will now be readied and Picture 6 will appear on the screen. Follow the instructions until you arrive at type of installation choice on Picture 8.

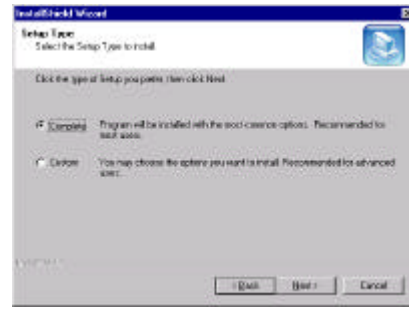


Picture 5 Installation Step 5

Picture 6 Install Step 6



Picture 7 Installation Step 7

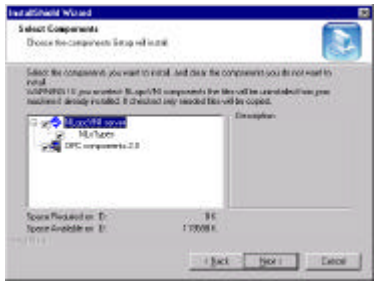


Picture 8 Install Step 8

You have the choice between the following installations:

Installation	***Details***
Complete	Complete installation of NLOPC-VNI and other generic files.
Custom	You can choose the module to be installed

Table 2 Type of installation



You should restart your PC at the end of the installation, according to the instructions

NETWORK INTERFACE

The network interface allows a physical link to be created between the PC and LonWorks network.

Valid interfaces

Type of interface	Maker	Connection	NSI	VNI
PCC10	Echelon	Slot PCMCIA	X	X
PCLTA10	Echelon	Slot ISA	X	X
PCLTA20	Echelon	Slot PCI	X	X
SLTA10	Echelon	Port RS232	X	
LonDongle	D&H gmbh	Parallel port	X	
LPP	Gesytec	Slot ISA	X	
LPC	Gesytec	Slot PCI	X	

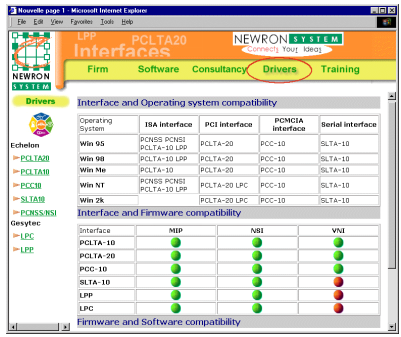
Table 3 Type of PC interface

To work, NLOPC-VNI needs a Firmware NSI or VNI interface

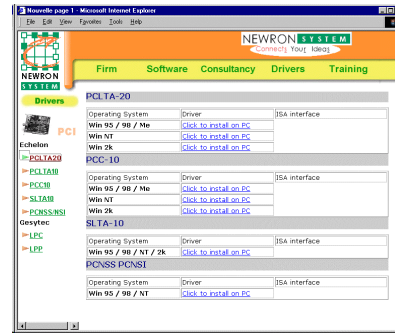
Driver for interface

Some drivers are available on CD on **Driver** main menu see: Picture 10.

Choose you network interface in the selection grid. Click on the appropriate interface in the vertical left menu. In Picture 11 we have selected PCLTA20. Choose the good operating system and follow instruction for setup.



Picture 10 Driver installation Step 1



Picture 11 Driver installation Step 2

Verifying interface function

To function correctly, NLOPC-VNI must have a type NSI or VNI interface as shown in Table 3.

Verifying PCC10, PCLTA10, PCLTA20

1. Open the configuration panel and launch the icon 

Verifying SLTA10

1. Activate: *Start/Programs/LonWorks SLTA10/SLTALink Manager*

2. The SLTA Link Manager menu bar must be as in Picture 12 with a green warning light, if it is red, or if the menu bar is like Picture 13, the interface is not working. Repeat the configuration steps, following the instructions carefully

Picture 12 SLTA10 connected

Picture 13 SLTA10 unconnected

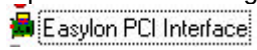
Verifying LonDongle

3. Plug the NLOPC Dongle in first, the LondonGle in second position.

4. Open the configuration panel and launch the icon «LondonGle Configuration Utility» 

5. Launch Diagnostics.. option.

Verifying LPP

1. Open the configuration panel and launch the icon 

2. If a check box is valid, the interface is show by the system like in Picture 14

Picture 14 Valid LPP windows

NLOPC VNI GENERAL

Introduction

NLOPC VNI is a LNS native OPC server. You need to have a LNS server installed on your machine.

This includes an OPC dataserver V2.0, it also handles the “object browsing” function.

This OPC server needs only to be configured. It can work with a single or several LNS database. You can also simulate LNS database.

LNS compliant

NLOPC VNI is LNS native. It supports to be executed with any other LNS application. For example it can run simultaneously with NL220, the LNS manager tool.

NLOPC VNI fully supports iLon and any EIA852 compliant IP router.

NLOPC VNI fully supports Configuration properties.

NLOPC VNI fully translates SNVT or SCPT, even unique field from a structure.

NLOPC VNI fully translates UNVT or UCPT, even unique field from a structure. This is available if you included the User type definition in the catalog file (Ressource file)

NLOPC VNI fully supports monitor points and monitor sets

Normal features

- ✓ Designed to be compliant to OPC 2.0 specification.
- ✓ Connects LonWorks networks to process control-, visualisation-, SCADA, and front office software
- ✓ Allows to tag LonWorks devices: nodes and routers
- ✓ Allows to tag LonWorks Network Variables
- ✓ Allows to tag LonWorks Configuration properties
- ✓ Allows to tag LonWorks Monitor points

- ✓ Runs under Windows 9x, Me, NT 4, 2000, XP

Advanced features

- ✓ Operate as client/server architecture:
- ✓ Supports COM and DCOM mode
- ✓ Multi-client compatible
- ✓ Can run in high level simulation mode in order to work offnet
- ✓ Can run with high-level trace mode to help you debugging your supervision.
- ✓ Supports OPC standard browser interface to access all items of its database.
- ✓ Supports several projects monitoring in a single OPC interface.

Network variable (NV) handling

NLOPCVNI uses three different ways to handle NV values:

1. Use of update events: the server mark the NV and it will wait for a network event.
2. Polling update events: the server marks the NV and it will receive an event to refresh its value on a certain frequency, even if the value has not changed.
3. Synchronous read: the server does not mark the item. The server will make itself the synchronous read at a defined frequency (slower)

Node handling

NLOPCVNI allows you to control the state of a node on the network.

Node handling uses a network variable as reference for the node.

If the referenced network variable pass in bad state, then the server considered that the module is in bad state.

Config property handling

NLOPCVNI allows you to control a config property on the network:

Synchronous read is the only way available to access config properties values.

Monitor point handling

A network variable that must be monitored can be referred to as a monitor point.

These items can use the same handling than network variables.

Formatted values:

NLOPC VNI allows you to handle SNVTs, SCPTs, UNVTs, UCPTs on network variable, configuration properties and Monitor points depending on the SNVT and SCPT Master List and the type files of LonWorks device templates.

It assumes all packaging, converting and updating tasks required for these objects. You can choose to get them in raw or in formatted value.

With NLOPCVNI you can access directly to a field of a structured SNVT, SCPT, UNVT, UCPT.

Limited version:

NLOPC VNI is available for test purpose in limited version.

NLOPC has no limited function: you can check its entire functionality.

In evaluation version NLOPCVNI is limited to:

- 20 minutes running.
- Only 10 NV write allowed.
- Only 50 items monitoring allowed.

A hard key unlocks NLOPCVNI (dongle).

Contact your distributor to get a hardware key.

Non standard Errors:

This is useful for an OPC client development:

ProgramID of NLOPC VNI is: **NewronSystem.NLopcVNI[.1]**

NLopc non standard errors are:

OPC_E_LONWRITE
(0xC0008001) Error writing an item on the network.

OPC_E_LONREAD1
(0xC0008002) Error reading an item on network.

OPC_E_LONREAD2
(0xC0008003) Error on a node state (.Module item).

OPC_E_SIMULATE
(0xC0008004) Cannot read the item because the driver is not opened.

OPC_E_DONGLE
(0xC0008005) Dongle hardware key is absent.

OPC_E_OPENDRVWITHLNS
(0xC0008006) Cannot open driver.

OPC_E_CANT_MODIFY_CONST_CP
(0xC0008009) Error on attempt to modify a constant configuration property.

NLOPC VNI CONFIGURATION

Introduction

NLOPC VNI needs to be configured. User needs to define a minimum of parameters, for example the LNS project he wants to work with.

User can define advanced options such as the alis, the browsers, …

General configuration (Mandatory)

Execute NLOPC configuration:

1. Activate: *Start / Programs / NlSuite / Nopc VNI / NLOPC VNI Configuration*

: Set the mode of NLOPC VNI:

Local: means you use a local LNS database in normal mode

Remote: you use a remote LNS database (over IP or LonTalk)

VNI: you use the VNI Interface for MonitorSet

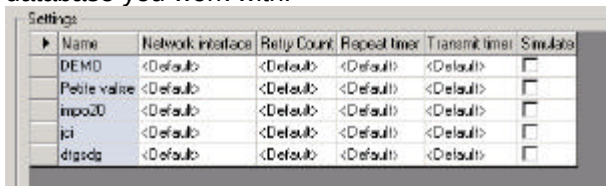
2. 

: Set the OPC mode compared to LNS:

All projects: NLOPC will propose items from several LNS database

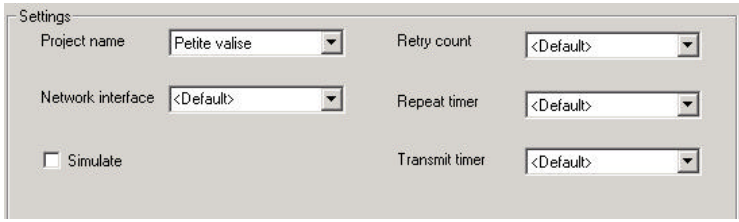
One project: NLOPC will propose items from one LNS database

3. When you choose ***All projects*** option, a table propose the entire LNS project in the database. You need to define the different LNS database you work with:



You can update the PC interface name, the timers and indicate if you want the OPC server to work in simulation mode on this project.

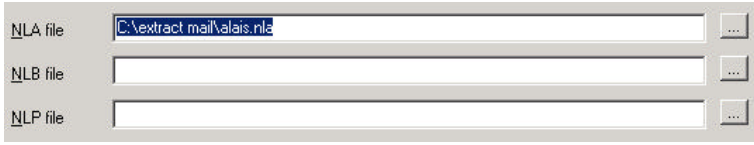
4. When you choose ***One project*** option, you need to define the LNS database you work with:



You can update the PC interface name, the timers and indicate if you want the OPC server to work in simulation mode on this project.

Advanced properties (Optional)

Optional extra files:

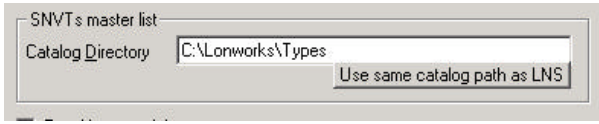


NLA files: you can provide an alias for any item name. In this case you access the item value from the OPC item tag name (cf) or from this alias. See the help file for NLA file format.

NLB files: you can define your own browser tree where OPC items are included. By default the OPC browser tree is: LNS database/Subsystems/nodename/LonmarkObject/Nvname. See the help file for NLB file format.

NLP files: when you monitor a node the string result is “absent” or “correct”. You can define a custom string for any node is the node is absent or not.. See the help file for NLP file format.

Catalog file path for SNVT automatic format:

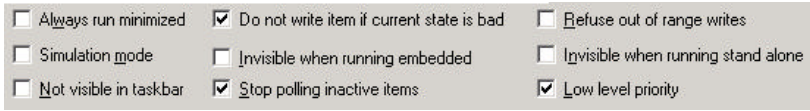


You provide the path of the catalog file. You have a button to use by default the LNS catalog path.

Read on init:

You have the ability to program the OPC server to read or not at init. If you do not read you can either return BAD or UNCERTAIN for the quality.

General options:



Always run minimized	if checked NLOPC will always be only in the windows toolbar while running.
Simulation mode	if checked NLOPC will run in simulation mode: it will not access to the network.
Not visible in task bar	if checked NLOPC will not appear even in the Windows taskbar. It will only appear in Task manager
Do not write items if current state is bad	Prevent from updating an item value if last reading on the device is not correct.
Invisible when running embedded	NLOPC is invisible in embedded mode (Launched from the OPC Client)
Stop polling inactive items	Disable LNS monitoring if a node is inactive
Refuse out of range writes	Check for value range and refuse to write if value is not correct
Invisible when running standalone	NLOPC is invisible in standalone mode (Launched from Windows)
Low level priority	OPC server tasks are launched in low priority mode

Settings and tools:

Conversion: Program NLOPC to automatically convert data. You program rules for each Nv. We provide SI to imperial conversion so that you can automatically get temperature in Fahrenheit for example. For more details see the help file

Enumeration: Program NLOPC to automatically convert enumeration of SNVTs. You program rules for each enumeration of each nv. You can automatically update ST_ON by ACTIVE for example. For more details see the help file

LNS2CSV: Utility that exports a LNS database and create a CSV file. For more details and to see the export parameters see the help file

OPC ITEM NAME

Introduction

NLOPC VNI proposes OPC items. An OPC item is the item you supervise. It can be Network variable or a node.

These OPC item can be extracted from a browser file or can be directly typed into the tag name of the supervision.

In all cases OPC item name follows rules.

Network variables

Formatting

A network variable can be:

1. a SNVT network variable.
2. a known UNVT network variable (it means a UNVT declared in catalog file)
3. an unknown UNVT network variable (it means a UNVT not found in the catalog file).

Then the value of the network variable can be accessed with several format called:

- Raw.
- Formatted.

The default format of a network variable item is:

for an enumeration: VT_I4.
for an integer value: VT_I4
for a float value: VT_R8
for any other type: VT_ARRAY|VT_UI1.

Raw address

The default format for a raw value is STRING (VT_BSTR).

A raw value can be cast in ARRAY OF BYTES (VT_UI1|VT_ARRAY) or in STRING (VT_BSTR)

The format of its address is:

NV,[<LNS Database name>::]<subsystem path>.<node name>.<network variable name>.Raw

with:

NV: Network variable prefix.

<LNS database name> Name of the LNS database which contains the network variable.

☞ Used only for server working on several projects.

Not used on simple project monitoring.

<subsystem path> Complete subsystem path of the device of the network variable in LNS database.

☞ Each subsystem folders are dot separated.

🛠 Ie: Location.Building1.Floor1

<node name> Name of the node of the network variable in the database.

<network variable name> Name of the network variable in the database.

Raw Suffix for raw format.

🛠 ie: When working on a simple project:

NV,Locations.Building1.Floor1.node1.nv1.Raw

When working on a several projects:

NV,MyProject::Locations.Building1.Floor1.node1.nv1.Raw

Formatted address

The default format for a formatted value depends of the type of the network variable or of the field:

for an Enumeration LONG (VT_I4)
For an Integer number LONG (VT_I4)
For a Float number DOUBLE (VT_R8)

The format of its address is:

NV,[<LNS Database name>::]<subsystem path>.<node name>.<network variable name>.field(<field name>)

with:

NV Network variable prefix.

<LNS database name> Name of the LNS database which contains the network variable.

☞ Used only for server working on several projects.

Not used on simple project monitoring.

<subsystem path> Complete subsystem path of the device of the network variable in LNS database.

☞ Each subsystem folders are dot separated.

🛠 Ie: Location.Building1.Floor1

<node name> Name of the node of the network variable in the database.

<network variable name> Name of the network variable in the database.

Field(<field name>) Name of the field in the type.

☞ If the type is a SNVT or UNVT without field (simple type) in this case it is the name than the SNVT or UNVT.

🛠 ie1: Simple type

SNVT_lev_disc does not contain any field.

When working on a simple project:

NV,Locations.Building1.Floor1.node1.nv1.field(SNVT_lev_disc)

When working on a several projects:

NV,MyProject::Locations.Building1.Floor1.node1.nv1.field(SNVT_lev_disc)

🛠 ie2: The SNVT_switch structure is:

```
struct {
    unsigned  value;
    signed  state;
 } SNVT_switch;
```

When working on a simple project:

🛠 NV,Locations.Building1.Floor1.node1.nv1.field(value)

or

🛠 NV,Locations.Building1.Floor1.node1.nv1.field(SNVT_switch.value)

When working on several projects:

NV,MyProject::Locations.Building1.Floor1.node1.nv1.field(value)

Or

NV,MyProject::Locations.Building1.Floor1.node1.nv1.field(SNVT_switch.value)

Casting to another standard type

You can cast any network variable to any SNVT type.

The only limit is that the casting type must have the same length as the native type.

ie: To format a SNVT_count_inc to a SNVT_count:

NV,Locations.Building1.Floor1.node1.nv1.field(SNVT_count)

Casting to another non standard type

You can cast any network variable to any UNVT type present in the catalog.

The only limit is that the casting type must have the same length as the native type.

ie: To format a SNVT_count_inc to a UNVT_mytype linked to the programid 9103041245223377

NV,Locations.Building1.Floor1.node1.nv1.field(#9103041245223377.UNVT_mytype)

Host network variable

Same as network variable (Cf Network variables p 19) You can get the raw value, formatted value, casting feature, …. but the OPC item prefix change.

You need to update the prefix **NV** by **HOST**

Raw address

The default format for a raw value is STRING (VT_BSTR).

A raw value can be cast in ARRAY OF BYTES (VT_UI1|VT_ARRAY) or in STRING (VT_BSTR)

The format of its address is:

HOST@[<LNS Database name>::]<Host interface name#><network variable name>.Raw

with:

HOST Host network variable prefix.

<LNS database name> Name of the LNS database which contains the network variable.

Used only for server working on several projects.

Not used on simple project monitoring.

<Host interface name#> Host interface name which contains the network variable.

Use if there are several network variable with the same name in the host.

Ie: Main interface

<network variable name> Name of the network variable in the database.

Raw Suffix for raw format.

🛠 ie: When working on a simple project:

HOST@Main interface#nv1.Raw or HOST@nv1.Raw

When working on a several projects:

HOST@MyProject::Main interface#nv1.Raw or
HOST@MyProject::nv1.Raw

Formatted address

The default format for a formatted value depends of the type of the network variable or of the field:

for an Enumeration LONG (VT_I4)
For an Integer number LONG (VT_I4)
For a Float number DOUBLE (VT_R8)

The format of its address is:

HOST@[<LNS Database name>::]<interface name>.<network variable name>.field(<field name>)

with:

HOST Host network variable prefix.

<LNS database name> Name of the LNS database which contains the network variable.

☞ Used only for server working on several projects.

Not used on simple project monitoring.

<interface name> Name of the interface on the host which contains the network variable.

☞ Use if there are several network variable with the same name in the host.

🛠 Ie: Main interface

<network variable name> Name of the network variable in the database.

Field(<field name>) Name of the field in the type.

If the type is a SNVT or UNVT without field (simple type) in this case it is the name of the SNVT or UNVT.

🛠 ie1: Simple type

SNVT_lev_disc does not contain any field.

🛠 When working on a simple project:

with the interface name:

HOST@Main interface#nv1.field(SNVT_lev_disc)

without specifying the interface name:

HOST@nv1.field(SNVT_lev_disc)

🛠 When working on a several projects:

with the interface name:

HOST@MyProject::Main interface#nv1.field(SNVT_lev_disc)

without specifying the interface name:

HOST@MyProject::nv1.field(SNVT_lev_disc)

🛠 ie2: The SNVT_switch structure is:

```
struct {
    unsigned  value;
    signed  state;
} SNVT_switch;
```

🛠 When working on a simple project:

with the interface name:

HOST@Main interface#nv1.field(value) or
HOST@Main interface#nv1.field(SNVT_switch.value)

Without specifying the interface name:

HOST@nv1.field(value) or
HOST@nv1.field(SNVT_switch.value)

When working on a several projects:

With the interface name:

HOST@MyProject::Main interface#nv1.field(value) or
HOST@MyProject::Main interface#nv1.field(SNVT_switch.value)

Without specifying the interface name:

HOST@MyProject::nv1.field(value) or
HOST@MyProject::nv1.field(SNVT_switch.value)

Casting to another standard type

You can cast any network variable to any SNVT type.

The only limit is that the casting type must have the same length as the native type.

ie: To format a SNVT_count_inc to a SNVT_count:

HOST@Main interface#nv1.field(SNVT_count)

Casting to non standard type

You can cast a host network variable to any UNVT type present in the catalog.

The only limit is that the casting type must have the same length as the native type.

ie: To format a SNVT_count_inc to a UNVT_mytype linked to the programID 9103041245223377

NV,Locations.Building1.Floor1.node1.nv1.field(#9103041245223377.UNVT_mytype)

Configuration parameters

Same as network variable (Cf Network variables p 19) You can get the raw value, formatted value, casting feature, …. but the OPC item prefix change.

You need to update the prefix **NV** by **NCP**

A configuration property can be:

- A SCPT configuration property.
- A known UCPT configuration property.(UCPT declared in catalog)
- An unknown UCPT configuration property (type not in catalog).

Then the value of the configuration property can be accessed through the NLOPCVNI server with several format called:

- Raw.
- Formatted.

Supervising configuration properties

Reading configuration properties is slow.

So it is not a good method to create a configuration item as active in an active group. This can dramatically slow down the performance of NLopcVNI.

If you do that set a large group frequency.

The better method to use configuration property is to read the item only using asynchronous or synchronous readings when needed.

You can:

Add the item in an inactive group
Set the item as inactive
Use the option /rq

Raw item address

The default format for a raw value is STRING (VT_BSTR).

A raw value can be cast in ARRAY OF BYTES (VT_UI1|VT_ARRAY) or in STRING (VT_BSTR)

For a node config property:

CPMODULE,[<LNS Database name>::]<subsystem path>.<node name>.<Config property name>.[<Element(Index)>].Raw

For a network variable config property:

CPNV,[<LNS Database name>::]<subsystem path>.<node name>.<network variable name>.<Config property name>.[<Element(Index)>].Raw

For a LonMark object config property:

CPLM,[<LNS Database name>::]<subsystem path>.<node name>.<LonMark object name>.<Config property name>.[<Element(Index)>].Raw

with:

CPMODULE	Prefix for node's configuration.
CPNV	Prefix for network variable's configuration
CPLM	Prefix for LonMark object configuration
<LNS database name>	Name of the LNS database which contains the network variable.

Used only for server working on several projects.

Not used on simple project monitoring.

<subsystem path>	Complete subsystem path of the device in LNS database.

☞ Each subsystem folders are dot separated.

🛠 Ie: Location.Building1.Floor1

<node name>	Name of the node in the database.
<network variable name>	Name of the network variable of the node. Only for CPNV..
<LonMark object name>	Name of the LonMark object of the node. Only for CPLM.
<Config property name>	Name of the configuration.
<Element(Index)>	Index of the element to be monitor in the config property item

Optional if the config property has only a single element.

raw	Suffix for raw format.

🛠 When working on a simple project:

For a node's config property:

CPMODULE,Locations.Building1.Floor1.node1.scpt1.Raw

For a NV config property:

CPNV,Locations.Building1.Floor1.node1.nv1.scpt1.Raw

For a LonMark object config property
CPLM,Locations.Building1.Floor1.node1.LM1.scpt1.Raw

When working on several projects:

For a node's config property:
CPMODULE,MyProject::Locations.Building1.Floor1.node1.scpt1.Raw

For a NV config property:
CPNV,MyProject::Locations.Building1.Floor1.node1.nv1.scpt1.Raw

For a LonMark object config property
CPLM,MyProject::Locations.Building1.Floor1.node1.LM1.scpt1.Raw

Formatted address

The default format for a formatted value depends of the type of the network variable or of the field:

for an Enumeration	LONG (VT_I4)
For an Integer number	LONG (VT_I4)
For a Float number	DOUBLE (VT_R8)

For a node config property:

CPMODULE,[<LNS Database name>::]<subsystem path>.<node name>.<Config property name>.[<Element(Index)>].Field(<fieldname>)

For a network variable config property:

CPNV,[<LNS Database name>::]<subsystem path>.<node name>.<network variable name>.<Config property name>.[<Element(Index)>].Field(<fieldname>)

For a LonMark object config property:

CPLM,[<LNS Database name>::]<subsystem path>.<node name>.<LonMark object name>.<Config property name>.[<Element(Index)>].Field(<fieldname>)

with:

CPMODULE	Prefix for node's configuration.
CPNV	Prefix for network variable's configuration
CPLM	Prefix for LonMark object configuration
<LNS database name>	Name of the LNS database which contains the network variable.

Used only for server working on several projects.

Not used on simple project monitoring.

<subsystem path>	Complete subsystem path of the device in LNS database.

☞ Each subsystem folders are dot separated.

🛠 Ie: Location.Building1.Floor1

<node name>	Name of the node in the database.

<network variable name> Name of the network variable of the node. Only for CPNV..

☞ Only for CPLM.

<LonMark object name> Name of the LonMark object of the node.

<Config property name> Name of the configuration.

<Element(Index)> Index of the element to be monitor in the config property item

Optional if the config property have only a single element.

Field(<field name>) Name of the field in the type of the configuration..

If the type is a SCPT or UCPT without field (simple type) in this case it is the name of the SCPT or UCPT.

ie1: Simple type

SCPTmaxSendTime does not contain any field.

🛠 When working on a simple project:

For a node's config property:
CPMODULE,Locations.Building1.Floor1.node1.scpt1.field(SCPTmaxSendTime)

For a NV config property:
CPNV,Locations.Building1.Floor1.node1.nv1.scpt1.field(SCPTmaxSendTime)

For a LonMark object config property:
CPLM,Locations.Building1.Floor1.node1.LM1.scpt1.field(SCPTmaxSendTime)

🛠 When working on a several projects:

For a node's config property:
CPMODULE,MyProject::Locations.Building1.Floor1.node1.scpt1.field(SCPTmaxSendTime)

For a NV config property:
CPNV,MyProject::Locations.Building1.Floor1.node1.nv1.scpt1.field(SCPTmaxSendTime)

For a LonMark object config property:
CPLM,MyProject::MyProject::Locations.Building1.Floor1.node1.LM1.scpt1.field(SCPTmaxSendTime)

ie2: The SCPTprimeVal structure is as the SNVT_switch structure:

```
  struct {
unsigned    value;
signed      state;
  } SCPTprimeVal;
```

When working on a simple project:

For a node's config property:
CPMODULE,Locations.Building1.Floor1.node1.scpt1.field(SCPTprimeVal.value)

or CPMODULE,Locations.Building1.Floor1.node1.scpt1.field(value)

For a NV config property:
CPNV,Locations.Building1.Floor1.node1.nv1.scpt1.field(SCPTprimeVal.value)

or CPNV,Locations.Building1.Floor1.node1.nv1.scpt1.field(value)

For a LonMark object config property:
CPLM,Locations.Building1.Floor1.node1.LM1.scpt1.field(SCPTprimeVal.value)

or CPLM,Locations.Building1.Floor1.node1.LM1.scpt1.field(value)

When working on a several projects:

For a node's config property:
CPMODULE,MyProject::Locations.Building1.Floor1.node1.scpt1.field(SCPTprimeVal.value)

or
CPMODULE,MyProject::Locations.Building1.Floor1.node1.scpt1.field(value)

For a NV config property:
CPNV,MyProject::Locations.Building1.Floor1.node1.nv1.scpt1.field(SCPTprimeVal.value)

or
CPNV,MyProject::Locations.Building1.Floor1.node1.nv1.scpt1.field(value)

For a LonMark object config property:
CPLM,MyProject::Locations.Building1.Floor1.node1.LM1.scpt1.field(SCPTprimeVal.value)

or
CPLM,MyProject::Locations.Building1.Floor1.node1.LM1.scpt1.field(value)

Casting to another standard type

You can cast any configuration property to any SCPT type.

For that the address format is the same with a <field name> different from the native type of the configuration property.

The only limit is that the casting type must have the same length as the native type.

ie:

For formatting a SCPTheatLowerSP to a SCPTheatUpperSP:

CPNV,Locations.Building1.Floor1.node1.nv1.scpt1.field(SCPTheatUpperSP)

Casting to another non standard type

You can cast any configuration property to any UCPT type present in the catalog.

For that the address format is the same with a <field name> different from the native type of the configuration property and with the program id (hexadecimal form) linked to the UCPT to use…

The only limit is that the casting type must have the same length as the native type.

ie:

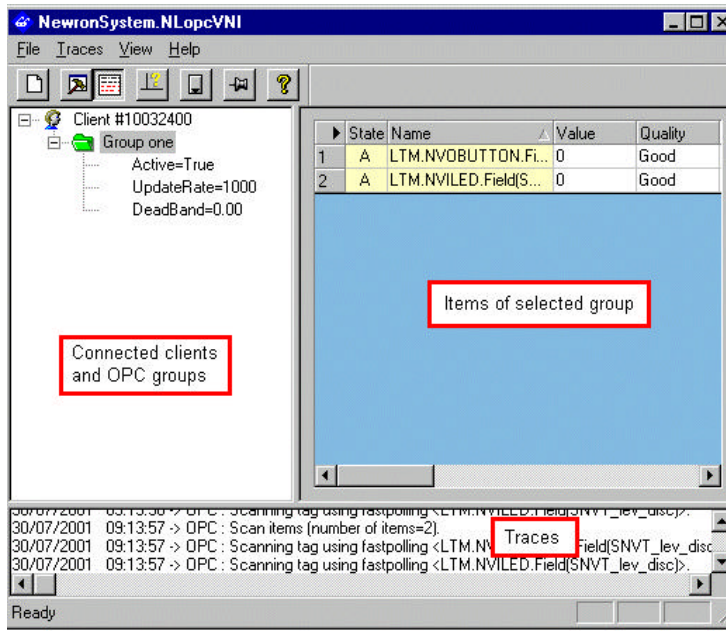
To format a SCPTheatUpperSP to a UCPT_mytype linked to the programid 9103041245223377

CPNV,Locations.Building1.Floor1.node1.nv1.scpt1.field(#9103041245223377.UCPT_mytype)

NLOPC USER INTERFACE

Human interface

NLOPC VNI is a full Windows program including 3 main views.



Picture 15
Human interface of NLOPC-VNI server

Three parts compose Nlopc user interface:

1. The upper left tree displays connected clients and OPC group created by each client.
   Selecting a group in this tree will display its contents in the right view.

   Each connected client is displayed with this icon .

   Each group is displayed with this icon ( ) if the group is active or with this one ( ) if inactive.
   In each group are displayed these informations:

Active	True if the group is active else false
UpdateRate	Update rate of the group. Unit is 1/1000 second.
Deadband	Dead band percent of the group.

Scan times is the time NLopc takes to refresh the entire group.

2. The upper right part of the screen will display the items of the selected group in the tree.

   This view is empty until you selected a group in the tree.

   Each items of the group is displayed with:

State	A for active, I for inactive
Value	Actual value of the item.
Quality	Actual quality of the item.
TimeStamp	Last time stamp.
Type	Type of the item
Dir	R for read only, R/W for read/write.

   You can filter the displayed items by name and/or by direction. For that right click on the grid and select option Filter tags from popup menu.

   The items are refreshed dynamically or manually. To manually refresh the items right click on the grid and select option Refresh items option.

   To configure the way the items are refreshed right click on the grid and select option Settings (see Configuring dynamic refresh topic)..

   In simulation mode you can double click on an item to change its value.

3. The lower part of the screen displayed all traces.

The toolbar buttons are:

 Clear traces

 Open traces settings window. See Trace options topic.

 Enable/disable traces. If pressed the traces are enabled.

 Open dynamic refresh settings window.

 Scroll down to end of traces and enable auto scroll.

 Set NLopc window always visible.

 Open About box.

Configuring dynamic refresh:

The clients, groups and group's items are dynamically refreshed on NLOPC screen. Dynamic refresh can use a lot of resources especially for dynamic items refresh.

You can configure the dynamic refresh by clicking on  button in the toolbar or by selecting the option Refresh settings in the View menu.

The options are:

Dynamic refresh add/remove tags If checked then new tags added or tags removed in/from the current selected group are dynamically refreshed.

Dynamic refresh tags value If checked then values qualities and timestamps of items are dynamically refreshed.

Real time refresh If checked, all changes will be refreshed immediately on screen.

Refresh every (seconds) If checked you can set the frequency (unit is second) to refresh the screen.

Remember that dynamic refresh is for debug purpose only.

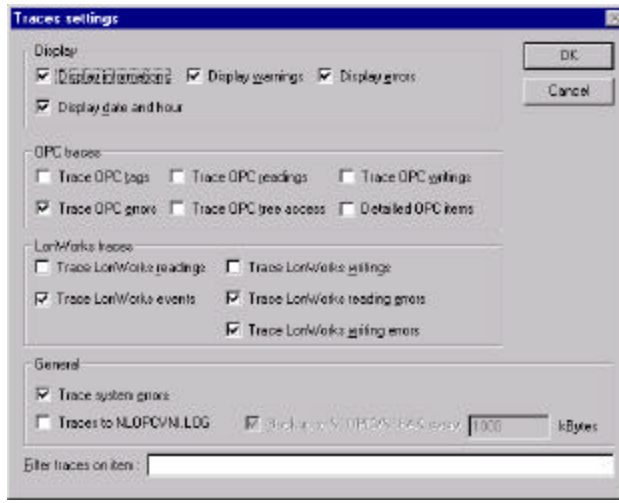
It can take a lot of CPU time especially with Dynamic refresh tags value and Real time refresh options.

Remember too that if no group is selected in the tree then NLopc will take no resources refreshing items.

The TRACE option:

NLOPC is provided with a trace option that allows you to debug very quickly your supervision. You have 2 kinds of trace: Trace on OPC or trace on LonWorks. Both traces can be logged in a file.

To set the settings for trace simply click on  in the toolbar. The



Picture 16
Windows traces settings

Display:

You choose what you want to display in the trace

Display information	Display or not information messages
Display warnings	Display or not warning messages
Display errors	Display or not error messages
Display date and hour	Display or not date and hour of event at the start of each line

OPC traces:

You can choose the kind of OPC trace you want to look for.

Trace OPC tags	Trace commands to add, remove and query tags
Trace OPC readings	Trace OPC groups scanning and OPC reads from client
Trace OPC writings	Trace OPC writes from client
Trace OPC errors	Trace OPC errors
Trace OPC tree access	Trace all tree accessing
Detailed OPC items	Display all items of a group when tracing group readings

LonWorks traces

You can choose the kind of LonWorks trace you want to look for.

Trace LonWorks reading	Trace all readings LonWorks messages
Trace LonWorks writings	Trace all writings LonWorks messages
Trace LonWorks bindings	Not used in this version
Trace LonWorks reading errors	Trace errors when reading on network
Trace LonWorks writing errors	Trace errors when writings on network

General:

You can choose to trace the system errors and also to log the traces in a file with a maximum size to avoid disk crash.

Trace system errors	Trace system errors
Trace to NLOPCVNI.LOG	Write all traces in NLOPCVNI.LOG. The file is created in same folder as NLOPCVNI.

AVOID KEEPING THIS OPTION CHECKED IN NORMAL MODE.

TRACES TO FILE SLOW DOWN PERFORMANCE OF NLOPC

Backup to NLOPCVNI.BAK every	If checked, NLOPCVNI will backup NLOPCVNI.LOG file on a give size and then will empty NLOPCVNI.LOG file.
kBytes	Size for creating NLOPCVNI.BAK

Filters trace on item:

You can filter the trace on items. The joker * is fully supported:

node1:	you will only trace tag called "node1"
no*	you will only trace tags beginning with "no"
*no	you will only trace tags ending with "no"
no	you will only trace tags including the string "no"

FIRST USE WITH A OPC CLIENT

Generic OPC client

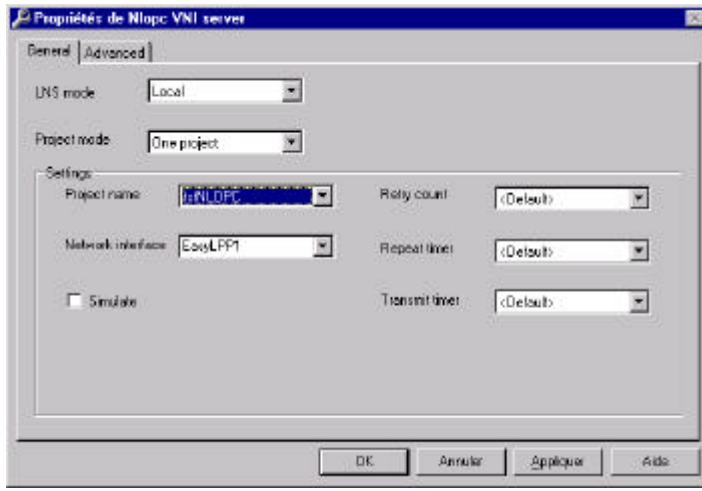
NLOPC is provided with a generic OPC client.

In this chapter we will use a LNS database with 1 node NL4EO.

Picture 17
LNS Database for example

STEP 1: Configure OPC server

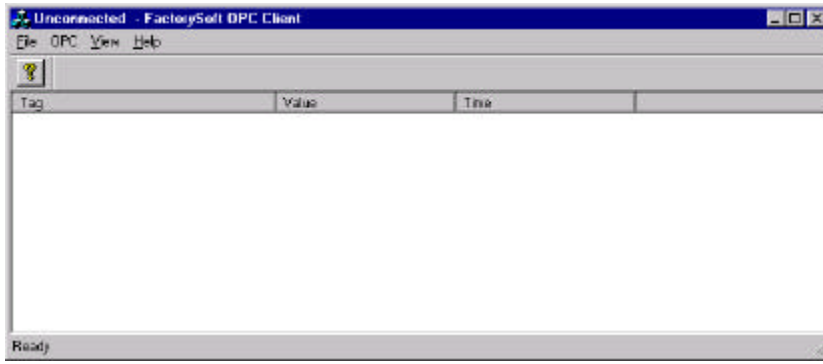
You need to parameter NLOPC to work with one or several LNS database (See Chapter NLOPC VNI configuration p 15)



Picture 18
NLOPC-VNI Configuration

STEP 2: Start the OPC client

Execute the OPC client. The following windows appear:



Picture 19
Main window on OPC Client

The OPC client is launched and you need to connect to an OPC Server. Choose the Menu option ***OPC***, ***Connect*** …

Picture 20
Select OPC Server

Choose the OPC server: ***NewronSystem.NLopcVNI.1***

And press OK to connect

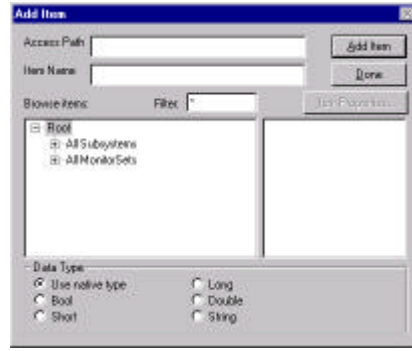
The OPC client will start NLOPC automatically: this is the embedded mode.

STEP 3: Browse OPC items

You are now connected to the OPC server. You need to add OPC item.

Browse the OPC server database: ***Menu OPC***, ***Add item***:

The list of internal tags of the OPC server is displayed:



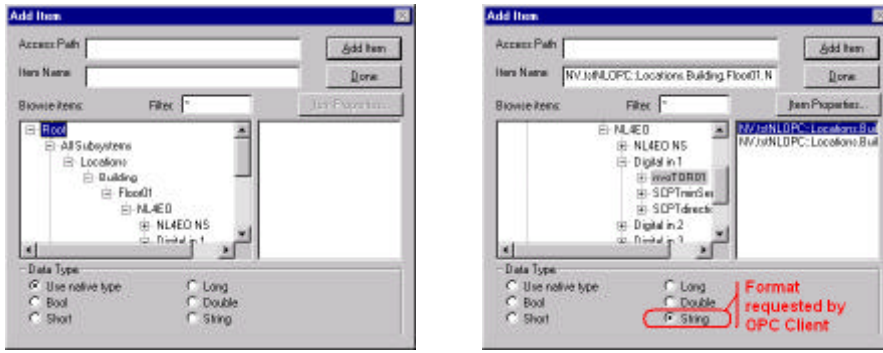
Picture 21
Browser closed in OPC Client

In **All Subsystems** tree you will browse the subsystems, then the nodes, then the LonMark objects then the network variables

In **All MonitorSets** tree you will browse the MonitorSets and then the Monitor points.

STEP 3: Add OPC items

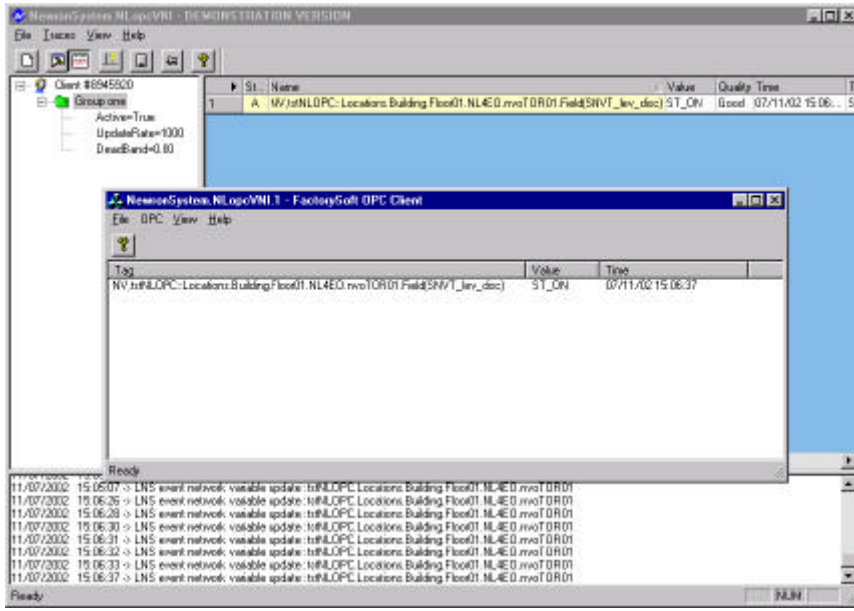
Browse the tree and choose an OPC item



Picture 22
Choose item on browser

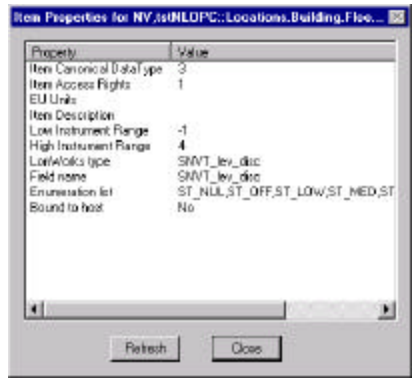
You can get the different OPC item name depending on OPC item format (Cf paragraph: OPC item name p19)

When you add this item, it is inserted in main window of OPC Client.



Picture 23
OPC Client and NLOPC-VNI with same item.

If you want extra information on the OPC item properties available in the OPC server, click on Item properties.



Picture 24
Item properties

Add all the items you need and then click on ***Done.***

The OPC client displays values.

STEP 4: Close OPC Client

Stop the OPC client application.

The OPC client will stop NLOPC automatically: this is the end of embedded mode.