NETx AUTOMATION

NETx BMS Server

First steps

Member of: KNX Association | OPC Foundation
BACnet Interest Group Europe

Document Version: 2.0.15

Contents

Copyright

This published handbook refers to the release of NETx BMS Server 2.0. The software is published by NETxAutomation Software GmbH, Maria–Theresia–Straße 41, Top 10, 4600 Wels, Austria.

© Copyright by NETxAutomation Software GmbH, 2015. The correct and usable documentation can only be guaranteed in connection with the regulations of the software agreement. Changes regarding the size of the function volume of the mentioned software can be done and may not involve a change of the documentation.

All rights are reserved. Copies, translations, micro filming and the storage and processing in data processing systems are copyrighted. No part of this publication may be reproduced without the prior permission of the publisher NETxAutomation Software GmbH.

1. Product Description

NETx BMS Server 2.0

The aim of the NETx BMS Server is to solve the problem that arises when heterogeneous building automation systems are used. To achieve this, the NETx BMS Server collects data and information from the field level of the building automation system using different fieldbus technologies. In NETx BMS Server 2.0 this data can originate from KNX, Modbus, or Building Automation and Control Networking Protocol (BACnet) networks. In addition, connections to other systems like Fidelio or to foreign systems that already provide an OPC connection are possible.

Once the data is available within the NETx BMS Server, management clients can access the data through the provided management interfaces. The NETx BMS Server provides interfaces to NETx Voyager clients, to Web-based NETx BMS Client clients as well as to any other third-party client. Thus, it builds a bridge between the field/automation level and the management level of building automation systems.

1.1. Main Criteria

- More than 100.000 datapoints are supported depending on the system's hardware
- Support for various fieldbus technologies (KNX, BACnet, Modbus, JSON)
- Multiple management client interfaces (OPC Clients, BMS Clients, or Web Clients)
- Interfaces to the NETx BMS Client visualizations and to Metering and Reporting System (MaRS)
- Server-based calendar for defining timer-based calendar events directly within the server
- Metering module for analyzing smart metering devices included
- Cluster module included that provides the opportunity to integrate foreign OPC DA and/or NETx BMS Servers.
- Interconnecting multiple NETx BMS Servers in a hierarchy using clustering.
- MS SQL database included that stores historical values of datapoints
- Nearly all officially used data types of KNX, BACnet, and Modbus are supported
- High reliability and availability
- Scales to large system – usable for smart homes up to large commercial buildings
- High data transfer rate
- Redundancy by hot standby main- and backup server configuration
- Device availability checking
- Multi-project kernel
- Workspace management – multiple workspaces can be administrated
- V-Links for implementing gateway functionality
- Virtual Items for specifying virtual datapoints
- Task definition table for linking of server items and executing LUA scripts in live mode
- LUA Script Language Engine for programming own control functionality with script editor
- Event Processor – cyclic, time and event based actions can be defined
- NETx BMS Server Studio 2.0
- Custom server items can be defined to add virtual datapoints
- Data recording in the queue buffer – up to one million sent or received telegrams
- Current state of the whole system can be saved and will be loaded automatically when starting
- Online check of the consistency of all connected gateways and devices
- Real-time view of the telegram traffic with clear text description and further information
- Geo Data Interface for sun position, sunrise, sunset, age of the moon
- Several extension modules available like SQL Interface for Microsoft SQL Databases, Micros Fidelio Hotel Information Interface ...

1.2. Disambiguation

1.2.1. Project/Workspace

A NETx BMS Server project contains all the information that is necessary to collect the data and information from the building automation systems. This includes on the one hand the configuration data that is required to get the data from the fieldbus systems and on the other hand runtime information like logging data. The sum of configuration and runtime files as well as the directories that include these files of a dedicated project are referred to as workspace. The workspaces are located within the following directory:

```
<Install Directory>\NETxAutomation\NETx.BMS.Server.2.0\Workspaces
```

For example, within Windows 8.1 64 Bit, the default path is:

```
C:\Program Files (x86)\NETxAutomation\NETx.BMS.Server.2.0\Workspaces
```

! It is recommended that the workspaces are backuped in regular intervals. To do so, the whole contents of the workspaces directory mentioned above has to be copied to the backup medium.

1.2.2. Datapoint

A datapoint is a single data element within the field level of the building automation system. This can be a sensor value (e.g. status of a light switch, temperature value), an actuator value (e.g. set value of a blind, control value of a heating system) but also so called virtual datapoints that are only present within the server (e.g. a set value for the room temperature, the current date, the current sun position).

1.2.3. Server Item

A server item is the representation of a datapoint within the NETx BMS Server. A server item consists of several properties that represent the datapoint. Important examples are the value of the datapoint, the quality as well as the engineering units. However, other meta-data may also be included (e.g. range of a datapoint, alarm status).

1.2.4. Server information model/Information model

The server information model is the internal view of the data and information of the building automation systems. In more detail, the server information model is organized as a hierarchy that consists of all server items as well as further information (e.g. type information).

1.2.5. Server Item Tree

The representation of the server information model is referred to as server item tree.

1.2.6. BMS Client

A BMS Client refers to a client application that connects to the NETx BMS Server to access the Server Items. A BMS Client can be NETx Voyager client, a NETx BMS Client, or any other third-party OPC client.

1.2.7. VNET

VNET is a proprietary network protocol that can be used by management clients as an alternative to OPC. VNET is available for all NETx management clients.

1.2.8. NETx BMS Client

A NETx BMS Client is a NETx client that uses either a HyperText Transfer Protocol (HTTP) connection (for Web-based clients) or a VNET connection (e.g. for WinCE clients) to get the data from the NETx BMS Server.

1.3. System Overview

Modern building automation systems are distributed systems where the control functionality is spread across a network. Due to the differing requirements of these systems, there is no single technology that can be used to satisfy all needs. As a result, building automation systems are extremely heterogeneous where many different network technologies and communication standards are used.

The aim of the NETx BMS Server is to solve this problem that arises when heterogeneous building automation systems are used. To achieve this, the NETx BMS Server collects data and information from the field level of the building automation system using different fieldbus technologies. In NETx BMS Server 2.0 this data can originate from KNX, Modbus, or BACnet networks. In addition, connections to other systems like Fidelio or to foreign systems that already provide an OPC connection are possible. For the latter one, the Cluster module has to be used that maps the OPC items from the third-party OPC server into the information model of the NETx BMS Server. Furthermore, due to the flexible and modular design of the NETx BMS Server, an integration of interfaces to other fieldbus technologies that are not available in the current version of the NETx BMS Server is possible. Figure 1.1 shows the basic architecture of the NETx BMS Server.

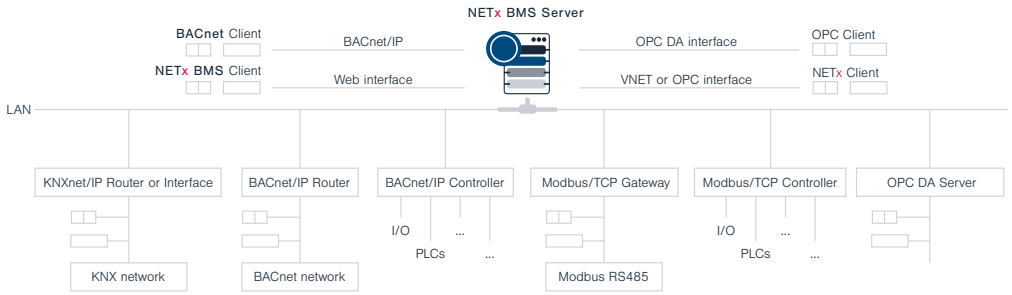

Figure 1.1.: System structure NETx BMS Server 2.0

1.3.1. System Architecture

The system consists of three main parts: the server, a Web server, and the NETx BMS Server Studio. The server is an autonomous application that implements core functionality. The Web server is service that is connected to the NETx BMS Server and provides the interface to the Web based clients. Finally, the NETx BMS Server Studio is the management interface that is used configure and maintain the server component and its workspaces. It includes useful tools which assists the administrator to managing the overall system including the NETx BMS Client clients and their visualization projects.

In Figure 1.2, the basic architecture of the overall NETx BMS Server system is shown.

A huge variety of integration is possible within NETx BMS Server 2.0.

From the field level, datapoints and devices from KNX, Modbus, and BACnet as well as from other systems like Micros Fidelio Hotel Information, Protel Hotel Information or VingCard door access system can be integrated within the NETx BMS Server. Furthermore, it is possible to integrate data from third-party OPC server using the Cluster module. The server retrieves all the information from the field level and stores it within its server information model. Within this model, the data is represented in a transparent, technological-independent way. This means that once the data is available as server item within NETx BMS Server, the underlying fieldbus technologies does not matter anymore – for a management client that accessed the information through the NETx BMS Server, the data is simply a server item and so it can handle it in a well-defined way. This has the aim that developers of management clients do not need to deal with characteristics of the underlying fieldbus technology.

Once the data is available through these generic server items, management clients can assess them through the provided management interfaces. The NETx BMS Server provided interfaces to the following management clients:

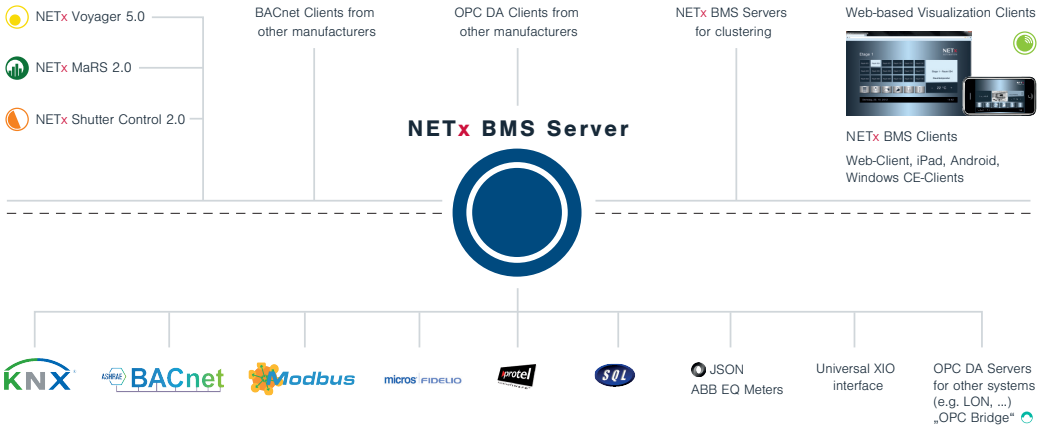

Figure 1.2.: System overview NETx BMS Server 2.0

- NETx Voyager clients that use OPC DA or VNET connection
- NETx BMS Clients that use a HTTP connection to the Web server or a VNET connection to communicate with the server
- Third-party OPC clients using OPC DA 2.05 connection
- Third-party BACnet clients using the BACnet Server interface

Depending on the operating system where a client is running, the following NETx BMS Clients are available:

- NETx Touch iOS Version: free client App for Apple's iPhone, iPad or iPod touch that can be installed via Apple's App Store.
- NETx Touch Android Version: free client App for Android devices (smart phones, tablets) that can be installed via Google Playstore.
- NETx Web Voyager Client is a client that uses the Web server of the NETx BMS Server for accessing the visualization. In fact any arbitrary Web browser supporting HTML 5 and JavaScript can be used – the used web engine within the NETx BMS Server does not required any browser plugins and so no additional software needs to be installed at the client side. Supported Web browsers are the commonly used ones.

Visualization projects for these clients are created and maintained within NETx BMS Server Studio using the so called NETx BMS Client Editor. This editor is the same for all three different kinds of NETx BMS Clients – therefore one visualization project can be used for all different client platforms at once. These clients are advantageous for smaller visualization projects where a large amount of NETx BMS Client devices have to be controlled and maintained centrally.

For large central visualization tasks, the NETx Voyager is available which can be connected via OPC or VNET. NETx Voyager is a PC-based, high-end visualization which provides more functionality that NETx BMS Clients. Typical examples are Alarm- and Event handling, task definition within the visualization, and scene control. NETx Voyager is a stand-alone application that has to be purchased separately.

1.3.2. The Server Item Tree

The basic principle of the NETx BMS Server is to get the data and information (i.e. datapoints) from the fieldbus and to store within the server as so called *server items*. Thus, a server item represents a single datapoint within the field level of the building automation system. Each server item consists of several properties that represent the datapoint within the building automation systems.

These server items are hierarchically arranged in a tree-like view called server item tree. This server item tree consists of the server items (leaves of the tree) as well as so called branches (nodes between the leaves and root of the tree) that are used to organize the tree. The server item tree can be seen as the network-visible representation of the server's information model that can be accessed by the management clients. Figure 1.3 shows an example of such a server item tree within the NETx BMS Server Studio.

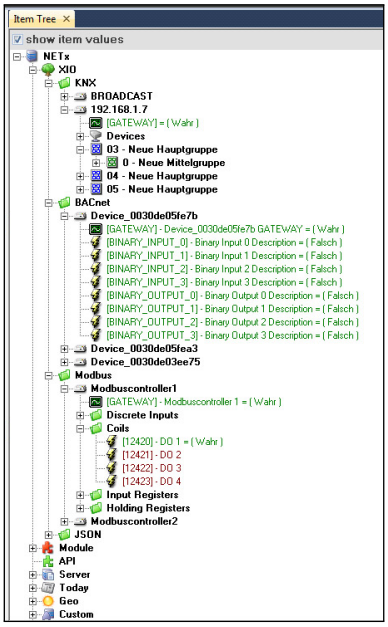

Figure 1.3.: Server Item Tree

The basic structure of this server item tree is unique within the NETx BMS Server. Server items and branches are addressed by clients using their item ID that is unique within the server item tree. This item ID refers to the browsing path that the item or branch has within the server item tree i.e. it is the concatenation of all item and branch names from the item or branch to the root node of the tree including the deliminators between the names[1]. The server item tree starts with the root node that has always the name:

`NETx\`

Within the first level of the server item tree, following branches are available:

`NETx\XIO\`

This sub-tree contains datapoints from the field level that are controlled by server. It has the following branches:

- `NETx\XIO\KNX\`: Here, the different KNX devices, router, and group addresses can be found. The following items are located within this sub-tree:
  - KNXnet/IP routers and interfaces: `NETx\XIO\KNX\<Router>\`
  - KNXnet/IP router and interfac monitoring item: `NETx\XIO\KNX\<Router>\GATEWAY\`
  - KNX group addresses: `NETx\XIO\KNX\<Router>\<GroupAddress>\`
  - KNX devices: `NETx\XIO\KNX\<Router>\Devices\IndividualAddress\`
- `NETx\XIO\Modbus\`: Here, the different Modbus devices and datapoints can be found. The following items are located within this sub-tree:
  - Modbus devices: `NETx\XIO\Modbus\<DeviceName>\`
  - Modbus devices monitoring item: `NETx\XIO\Modbus\<DeviceName>\GATEWAY\`
  - Modbus discrete inputs: `NETx\XIO\Modbus\<DeviceName>\Discrete Inputs\<Address>\`
  - Modbus coils: `NETx\XIO\Modbus\<DeviceName>\Coils\<Address>\`
  - Modbus input registers: `NETx\XIO\Modbus\<DeviceName>\Input Registers\<Address>\`
  - Modbus holding registers: `NETx\XIO\Modbus\<DeviceName>\Holding Registers\<Address>\`
- `NETx\XIO\BACnet\`: Here, the different BACnet devices and objects can be found. The following items are located within this sub-tree:
  - BACnet/IP devices: `NETx\XIO\BACnet\<DeviceName>\`
  - BACnet/IP routers: `NETx\XIO\BACnet\<RouterName>\`
  - BACnet non-IP devices: `NETx\XIO\BACnet\<RouterName>\<DeviceName>\`
  - BACnet devices monitoring item: `NETx\XIO\BACnet\...\GATEWAY\`
  - BACnet objects: `NETx\XIO\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\`
    For read-only objects:

[1] The deliminator can be configured within the server. For the rest of this documentation the default deliminator "\" is used.

...\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\<DatapointValue>\
For writable objects:
Reading:...\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\<DatapointValue>\
Writing:...\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\<DatapointValue>-PRIO <nr1>\
Resetting:...\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\<DatapointValue>-PRIO <nr1> RESET\
Writing:...\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\<DatapointValue>-PRIO <nr2>\
Resetting:...\BACnet\{<RouterName>}\<DeviceName>\<ObjectName>\<DatapointValue>-PRIO <nr2> RESET\
...

- NETx\XIO\JSON\: Here, the different JSON devices and datapoints can be found. The following items are located within this sub-tree:
  - JSON gateways: NETx\XIO\JSON\<GatewayName>\
  - JSON gateway monitoring item: NETx\XIO\JSON\<GatewayName>\GATEWAY\
  - ABB EQ Meter: NETx\XIO\JSON\<GatewayName>\<Serial>\
  - ABB EQ Meter monitoring item: NETx\XIO\JSON\<GatewayName>\<Serial>\GATEWAY\
  - ABB EQ Meter energy values:
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\importenergy\
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\importenergy\total\
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\importenergy\l[1-3]\
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\importenergy\t[1-4]\
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\exportenergy\
    ...
    NETx\XIO\JSON\<GatewayName>\<Serial>\apparent\netenergy\
    ...
    NETx\XIO\JSON\<GatewayName>\<Serial>\reactive\
    ...
    NETx\XIO\JSON\<GatewayName>\<Serial>\active\
    ...

NETx\Cluster\
Within this sub-tree, items of the Cluster module (e.g. items from foreign OPC DA Servers, items from other NETx BMS Servers) are located.

- NETx\Cluster\VNET\
  Here, the different Server Items of integrated, remote NETx BMS Servers can be found. For each NETx BMS Server, the sub-tree NETx\Cluster\VNET\<IPAddress>\<ServerName>\ is created. The following items are located within this sub-tree:
  - NETx\Cluster\VNET\<IPAddress>\<ServerName>\Connected\: This item indicates whether the connection to the NETx BMS Server is working.
  - NETx\Cluster\VNET\<IPAddress>\<ServerName>\<ItemName>\: In addition, all Server Item that shall be integrated from the remote NETx BMS Server are listed.
- NETx\Cluster\OPC_DA\
  Here, the different OPC Items of foreign OPC DA Servers can be found. For each OPC DA Server, the sub-tree NETx\cluster\OPC_DA\<IPAddress>\<ServerName>\ is created. The following items are located within this sub-tree:
  - NETx\Cluster\OPC_DA\<IPAddress>\<ServerName>\Connected\: This item indicates whether the connection to the OPC DA Server is working.
  - NETx\Cluster\OPC_DA\<IPAddress>\<ServerName>\<ItemName>\: In addition, all OPC items that shall be integrated from the foreign OPC DA Server are listed.

NETx\Module\
Within this sub-tree, items of NETx BMS Server modules are located. Currently, the following module is available:

- NETx\Module\MaRS\
  Here, the different resources and meters of the Metering module can be defined.

NETx\API\
Within this sub-tree, items of additional driver modules (e.g. items from the Fidelio module) are located.

`NETx\Server\`
Here, information about server is stored (e.g. status of server, number of connected clients, . . .). Within the sub-tree `NETx\Server\Database\` the status of database connection can be seen.

`NETx\Today\`
Within this sub-tree, time and date as well as weather information can be retrieved.

`NETx\Geo\`
Here, geographic information like sunset time and moon age is available.

`NETx\Custom\`
Using LUA scripts, it is possible to create so called custom items that can be seen as virtual items that are only available within the server. It is also possible to define them within a user-specific hierarchy. These custom items are located within this sub-tree.

`NETx\Aliases\`
It is possible to define aliases i.e. alternative names for server items. These aliases can be found here.

`NETx\VAR\`
This sub-tree contains pre-definied, virtual datapoints that can freely be used by clients.

`NETx\XCOMMAND\`
This sub-tree contains the definition of the so called XCOMMANDs that are used by the server-based event modules (for future use).

`NETx\VIRTUAL\`
Here, the virtual items that are defined within the current workspace are located. According to the configuration, these items can be arranged within a hierarchy.

`NETx\XCON\`
Here, different items that can be used for communication with other systems or network services are located (e.g. items for UDP, TCP, HTTP, COM port, e mail communication).

2. Getting Started

2.1. Installation

Insert the disk into the optical drive.

Either the installation software will start at autorun or the user will need to execute it manually.

The setup will guide the user through the installation. During this process it is possible to make a few specifications such as a different installation path than the default one.

Inside the directory – either specified by the user or the default one – a sub folder “Workspaces” will be created. The latter one will not be deleted when deinstalling the software. It has a special role in saving workspaces and also contains the default workspace, from which all new workspaces will derive.

! Note that under Microsoft Windows™ server systems the installation of Microsoft .NET Framework 3.5™ might need to be done manually by installing the according server feature.

!Attention: All anti virus software and anti mall ware has to be deactivated during installation. The user installing the software has to have appropriate rights to do so. Make sure the necessary exceptions are set in the security software so that NETx BMS Server is working properly. For compatibility reasons it is also not recommended to run NETx BMS Server parallel to NETx Voyager Server 1.0 on the same machine.

With the NETx BMS Server several different components are installed and registered on the system – the NETx BMS Server itself as a service or server component, the web server as a service to provide NETx BMS Clients with the necessary information, the NETx BMS Client Editor as an executable program, the NETx BMS Server Studio 2.0 as an executable program and different drivers to support BACnet, Modbus, KNX, and VNet as communication interfaces. Microsoft SQL Server Express™ is also installed to store historical data.

2.2. First Steps

First start the NETx BMS Server Studio 2.0. This can be accomplished by either clicking the icon on the desktop or one can run the program out of the start menu. When the software is run the first time and there is no valid hard-lock (USB stick) connected to the system, a warning will appear in the system messages that this is an unlicensed copy of the software and the demo mode will continue to work. To license the software either a valid hardware lock can be inserted or the License Manager has to be run. For first steps the evaluation mode is sufficient. For more detailed information about licensing the software, please refer to section A.2.

Again assuming the software is run the first time it will start the “MyFirstWorkspace” or the “Default” workspace. Once a new one is created and the program is closed, it will reopen with the last active one at startup.

! It is not recommended to work in the “Default” workspace since all future workspaces created will use it as template. All properties set in it will be inherited by the new workspace.

To create a new workspace, go to the menu “Workspace” and choose “New Workspace”. The program will now prompt the user with a mask to put in the new workspace name. Since the workspace name will also be used as folder name for the workspace it is not allowed to use special characters like quotation marks, slashes, backslashes, or similar.

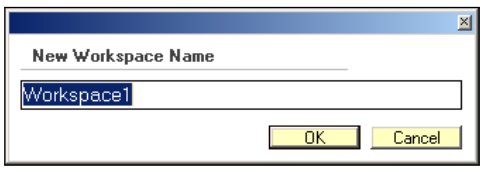

Figure 2.1.: Name new workspace

Should the server have not been shut down already, it will shut down now and an attempt will be made to start the newly created workspace. It contains a small demo project which is present in the “Default” workspace.

The next step to get the system up and running would be to include the configuration data of the devices and datapoints that are used at the field level of the building automation system.

2.2.1. KNX Configuration

The import for KNX group addresses is a quick way to add many group addresses at once. Open the menu item “KNX” and select “Import ETS© Project. . . ”.

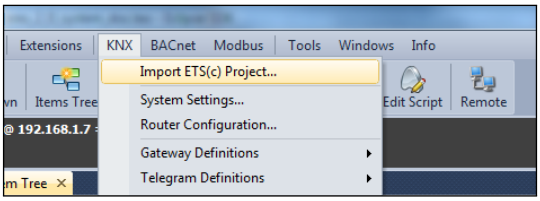

Figure 2.2.: Open ETS Converter

A dialogue will appear (cf. Figure 2.3) where on first line the file to import can be selected. The second important field is “Default Gateway” which determines the gateway the telegrams will be sent to for all the group addresses which will be imported this time. For the first steps it is suggested to type in the IP address of the KNXnet/IP router e.g. “192.168.1.70”. Check the boxes to create a telegram definition file, link definition file, and a gateway definition file and leave all other entries on their default values. Click “Convert” and after confirming the creation of the definition files close the mask.

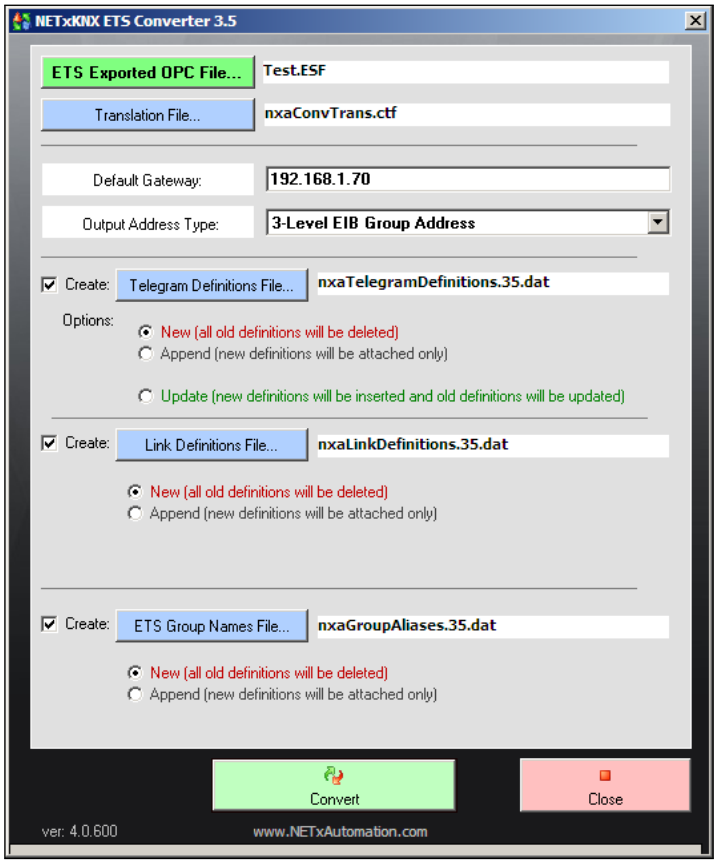

Figure 2.3.: Convert ETS File

2.2.2. Modbus Configuration

If there are Modbus devices connected to the system please follow the next steps. First open the device definition

table under the menu item “Device Definitions” within the “Modbus” menu (cf. Figure 2.4).

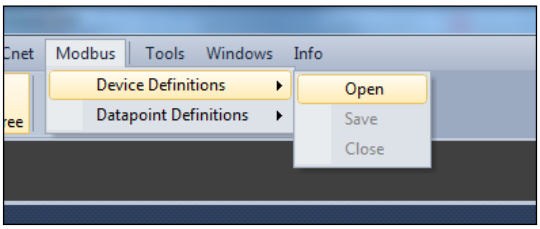

Figure 2.4.: Open Modbus Device Definition

Now either edit a line in it or add a new one using the according command of the context menu (cf. Figure 2.5). The first field “Device Name” is a unique name which can be freely chosen, the “IP Address” of the device needs to be entered in the second field and the “Port” to connect to in the third one. If the device is a native Modbus/TCP device “0” has to entered within the next field – otherwise the “Slave ID” has to be typed in. In the fifth field a description for the device can be added. All following fields are optional and therefore will be described in detail in a later chapter.

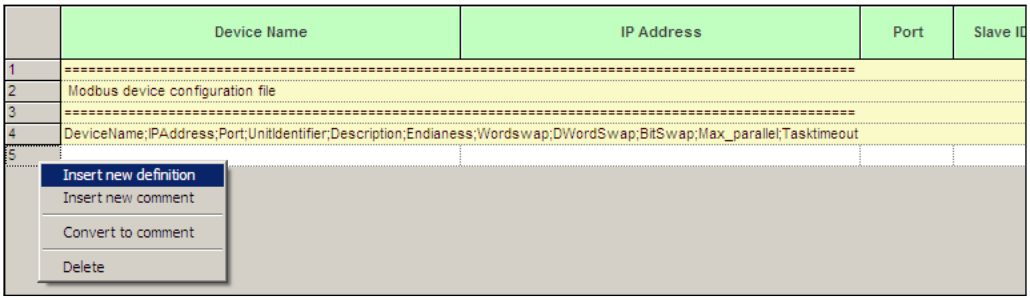

Figure 2.5.: Edit Modbus Device Definition

For editing the actual datapoints please open the according definition file (menu “Modbus”, menu item “Datapoint Definitions”). The “Device Name” needs to be identical to the one which has been defined previously. It represents the reference to the device that holds the actual datapoint. “Modbus DP Type” defines the datapoint type inside the Modbus device. The appropriate one needs to be selected. “Address” contains the memory address within the Modbus device where the datapoint is situated. “Data Type” defines which kind of data value the memory cell in the Modbus device contains. Should the data type be a “String” or “WString”, “Size” determines the length of it. All other data types define their length on their own and the field should be set to “1”. Enter a short “Description” of the datapoint. “Polling Interval” contains the value in milliseconds of the interval the data will be requested in from the device. Again all further fields are optional and therefore will be described in detail in a later chapter.

2.2.3. BACnet Configuration

The quickest way to include a correct BACnet configuration is to run the BACnet Explorer and save its output to the definition files.

If the BACnet Explorer is run the first time the user will be prompted with the message that “127.0.0.1” is not a valid address and that it should be changed. For now “Yes” would be a good choice. Now please select the correct IP address using the next dialogue. Confirm with “OK” and the actual BACnet Explorer will open and scan the network for available devices.

Check any of the device’s boxes and the datapoints will be scanned and included into the configuration. Click the “Export” button to write the data to the configuration files. After confirming the creation of the files the user can close the BACnet Explorer with the “Exit” button.

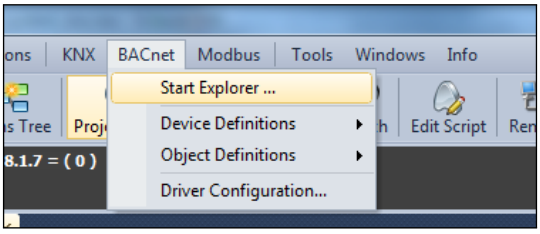

Figure 2.6.: Open BACnet Explorer

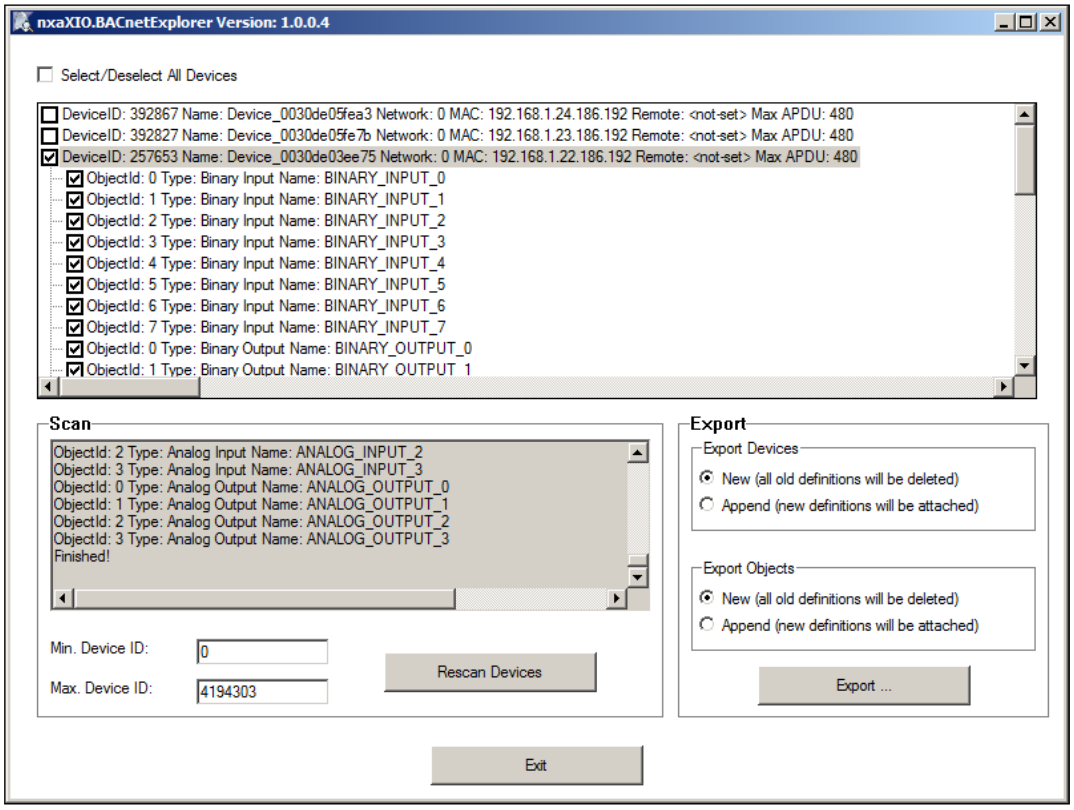

Figure 2.7.: BACnet Explorer

2.2.4. Starting the Server

To start the server click the green arrow in the toolbar. Some log information will be shown in the “System Messages” and then the “Item Tree” shows up. Under node “XIO” the devices can be found in their according category. When the devices there are opened, the datapoints that are assigned to the devices are shown. Using the menu item “Send Telegramm . . . ” or “Write Item Value . . . ” within the context menu (right-click on the datapoint), the values of the datapoints can be changed when they are writable. In addition, the current datapoint value and the corresponding item properties can be shown within “Properties” window at the right hand side.

2.2.5. Visualization Project

To create a new visualization project, right click “Visualization Project” inside the “Project Tree” and Choose “New visualization project. . . ”.

The user will be asked to put in the name of the new NETx BMS Client project. Afterward the NETx BMS Client Editor will open and the user will be able to design and edit the NETx BMS Client project.

Once the NETx BMS Client Editor is closed the project can be reopened for editing by either double-clicking the project’s name in the “Project Tree” or by the context menu’s item “Edit visualization project. . . ”. Once changes

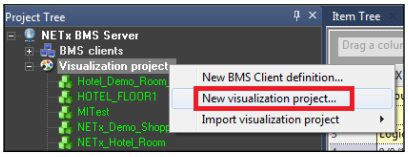

Figure 2.8.: Create a new Visualization Project

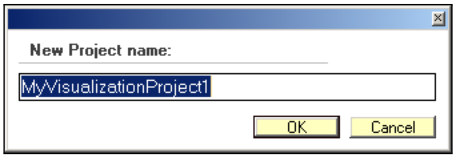

Figure 2.9.: Name the new visualization project

have been made to the project, it is necessary to push the project to the NETx BMS Clients. This can be achieved by the menu item “Push visualization project to BMS Clients. . . ” from either the context menu or the menubar / “Project”. Visualization projects shown greyed in the project tree do not have a linked workspace in the NETx BMS Client Editor and therefore have to be updated manually.

2.2.6. BMS Client

The NETx BMS Clients can be created very easily. A right-click to “BMS Clients” within the “Project Tree” will open a context menu from which “New BMS Client definition. . . ” can be chosen.

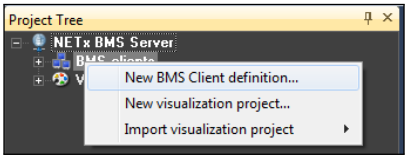

Figure 2.10.: Create new BMS Client definition

A dialog will appear and the specifications to the BMS Client can be made. The field “Name” has to be filled in to name the client. Set “Enabled” to true to be able to connect to the client. “Project” determines which of the NETx BMS Client projects shall be displayed with this BMS Client. Other fields are not mandatory and can be left empty.

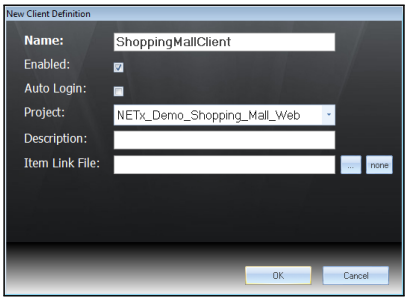

Figure 2.11.: BMS Client Properties

A. Appendix

A.1. Acronyms

BACnet Building Automation and Control Networking Protocol

HTTP HyperText Transfer Protocol

MaRS Metering and Reporting System

USB Universal Serial Bus

A.2. Licensing

Two different possibilites are available:

- Hardlock – Hardware based (Universal Serial Bus (USB) Dongle) security system
- Softlock – Software based security system

A.2.1. Hardlock

One of the solutions is hardlock. Just insert the USB-Dongle into any USB port in your system and let the hardware driver get installed. The software will automatically recognize and read your license from the USB dongle. It might be necessary to restart the software to make it recognize the license.

! The USB dongle has to be connected at all times from start of the application on. If it is disconnected the correct work of the software will stop after two warnings (which is approximately after 15 minutes).

A.2.2. Softlock

In this case the software license is assigned to a local code that depends on a checksum of the local hard- and software.

! The local code of the soft lock license can be changed by any major change of hardware (e.g. network adapter) or software (e.g. operating system). After such change it might be necessary to license the software again.

A.2.2.1. Software Licensing

The licensing process can be initiated by starting the “License Manager” from the Windows start menu (“NETxAutomation”, “NETx BMS Server 2.0”, “NETx Registration”).

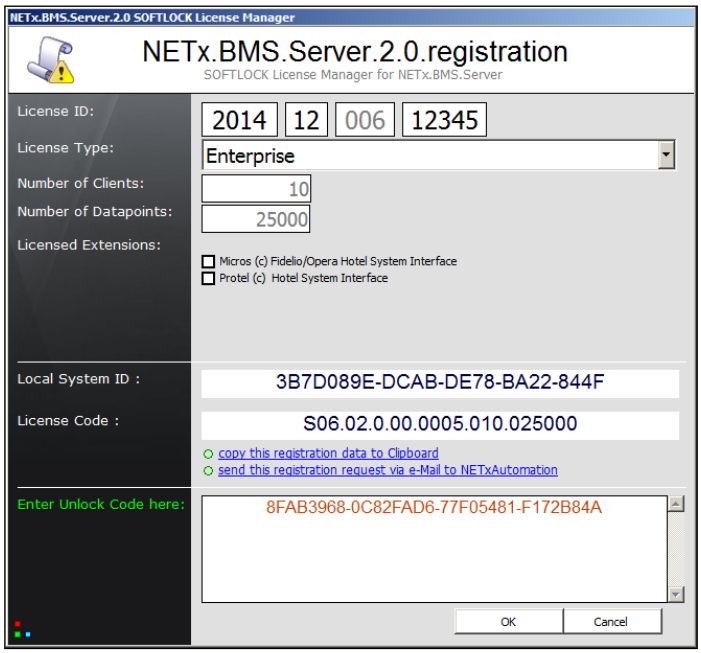

Figure A.1.: Licensing Software

The License Manager will show up with several fields:

- License ID – This block consists of 4 fields. Please fill in the License ID from the delivered invoice.

- License Type – Select the type according to the delivered invoice here. Amount of clients and amount of datapoints will depend on it. If none of the predefined licenses matches select “Custom” and enter the amount of clients and datapoints.
- Number of Clients – This field defines the number of clients to be licensed. For custom licenses, please enter the amount of clients here. If the order was a predefined package this field is not editable but will show the according number of clients.
- Number of Datapoints – This field defines the number of datapoints to be licensed. For custom licenses, please enter the amount of datapoints here. If the ordered was a predefined package this field is not editable but will show the according number of datapoints.
- Licensed Extensions – Check the extensions which have been ordered in here.
- Local System ID – This is the local system key. It is automatically generated by the “License Manager”.
- License Code – This code contains most of the information above. It is generated automatically.

Once the above listed fields are all filled out with the correct information send an email directly to NETxAutomation Software GmbH by clicking the according link. If there is no access to the Internet, choose the link “copy to clipboard”. Open a text file in which the clipboard content needs to be inserted (usually via [Ctrl]+[V] or “Edit” and “Paste”). Now send it from any other computer by mail to register@netxautomation.com.

The following entries must be sent to license the software:

- License ID
- License Type
- License Code
- Local Code
- Date
- Software Version

Transmit this data to NETxAutomation Software GmbH to receive the “Unlock Code”. Once it was received the unlock code must be inserted into the bottom field and confirmed with “OK”. It is to be made sure to copy the “Unlock Code” without any blanks, tabs or other white spaces. If an error message is received after the process, check the given data and try the registration once more.

! The user executing the licensing software must be local administrator. The program needs to be started as administrator.

A.2.2.2. Move a license

When it is planned to move the NETx BMS Server 2.0 from one system to another or the system will change (hardware upgrade), it is useful to unlicense the software. The unlicensing process can be initiated by starting the “License Manager ...” from the Windows start menu (“NETxAutomation”, “NETx BMS Server 2.0”, “NETx Registration”)

Press the “Transfer/Remove License” button to gain the removal code. Send this code to NETxAutomation Software GmbH to receive the new “Unlock Code” to license the software on the new or upgraded system.

A.2.3. License Count

The NETx BMS Server is licensed after datapoints and BMS Clients. In addition to regular(physical) datapoints the license includes always an equally high amount of virtual datapoints. Datapoints are all bus layer objects defined in tables and listed either in the XIO branch of the Item Tree or inside the Aliases. Virtual datapoints are considered all server items which have no direct relation to a physical device. They are either calculated items or items with a value brought in by an alien software (e.g. NETx OPC Bridge). Not all virtual items count into the license limit.

The following three lists define exactly which items count where:

Datapoints (regular/physical)

- KNX Items

- BACnet Items – only the BACnet Object itself is counted as one item. The items underneath representing the priorities are not counted.
- Modbus Items
- JSON Items
- Aliases – since items with an alias are rather moved into this branch than copied, the aliases will count but original items remain uncounted.

Virtual Datapoints (counted)

- All additional items created by in-/out-conversion
- All items in the branch NETx\Custom (added by LUA nxa.AddCustomItem())
- All items in the branch NETx\Module
- All items in the branch NETx\VIRTUAL (added by definition in the table)
- All items in the branch NETx\XCOMMAND
- All items in the branch NETx\XCON

Virtual Datapoints (not counted)

- All items in the branch NETx\API (e.g. the Fidelio interface)
- All items in the branch NETx\Server
- All items in the branch NETx\Today
- All items in the branch NETx\Geo
- All items in the branch NETx\VAR – all predefined variables

A.3. Support and contact

Please send all your support questions to:
support@NETxAutomation.com
If you have general questions regarding the product and service please send your email to:
office@NETxAutomation.com

A.4. System Requirements

A.4.1. Hardware

- Processor: Intel or AMD 1.6GHz (Multicore recommended)
- System Memory: 2048MB
- Harddisk Space: 8GB (16GB recommended)
- Network Adapter: 100 MBit/s
- Screen Resolution: 1280 x 1024 Pixel (for BMS Studio)

A.4.2. Supported Operating Systems

- Microsoft Windows 7 32bit Servicepack 1
- Microsoft Windows 7 64bit Servicepack 1
- Microsoft Windows 8 64bit
- Microsoft Windows 8.1 64bit
- Microsoft Windows Server 2008 32bit Servicepack 2
- Microsoft Windows Server 2008 64bit Servicepack 2
- Microsoft Windows Server 2008 Release 2 64bit Servicepack 1
- Microsoft Windows Server 2012 64bit
- Microsoft Windows Server 2012 R2 64bit

A.4.3. Other

- .NET Framework: 3.5 (included in setup)
- Microsoft Visual Studio Redistributable 2008 32 or 64 bit (included in setup)
- Microsoft Visual Studio Redistributable 2010 32 bit (included in setup)
- Microsoft Access Redistributable 2010 32 bit (included in setup)