

1 Abstract

Interfacing KNX installations via EIBnet/IP with building management or other SCADA systems is fast becoming a standard.

Engineering considerations for these systems demand a simple yet powerful, implementation and operating system-independent representation of characteristic data of a KNX installation. The eXtensible Markup Language (XML) fulfills these requirements and has thus been used as a franca lingua for system data interchange.

The presentation will lay out the requirements from the field and how these are covered by the XML encoding of KNX installation data defined by the KNX Task Force IP.

2 Introduction

The seamless integration of local and world-wide networks has opened new venues for the Building Electronic System. Even globally distributed facilities can be monitored from a central location around the clock.

EIBnet/IP, a consistent open multi-vendor standard conceived by Konnex Association, provides remote configuration, operation, and fast communication between EIB/KNX lines and installations. In principal the EIBnet/IP standard describes two communication types: Tunneling and Routing.

Routing allows for the propagation of EIB/KNX telegrams from one EIBnet/IP router to other EIBnet/IP routers. This is the basis for fast communication between lines, areas, or complete installations. A device implementing EIBnet/IP Routing is in effect a line coupler.

Tunneling enables point-to-point communication between two EIBnet/IP devices. This not only replaces the serial interface communication but also provides faster access to the EIBnet/IP installation from anywhere on the IP network.

Finally, the EIBnet/IP protocol defines Device Management for configuration of EIBnet/IP devices over IP networks.

The currently standardized EIBnet/IP protocol leads the Building Electronic System onto the data highway. More and more building owners and facility managers are taking advantage of this new KNX system capability, which is opening new application areas and market growth.

3 The Challenge

The currently standardized EIBnet/IP protocol provides Tunneling as an online interface protocol between KNX and other systems. This addresses the needs for interfacing to supervisory or SCADA systems, providing visualization, data capturing, email notification and other central system features.

As the standardized EIBnet/IP protocol offers faster interchange of more data than with the traditionally used serial interfaces, this increased number of data points has to be efficiently engineered in communication interfaces with supervisory systems. The necessary information has to be extracted from project documentation or, more likely, the ETS project. ETS2 and ETS3 both provide data export functions. Yet, these functions are different from each other and do not provide all necessary data for simple engineering.

The KNX Task Force IP has identified other potential application areas for EIBnet/IP like Remote Logging or Remote Configuration.

In a Remote Logging situation a KNX device captures data from the KNX subnetwork according to specific rules and stores this information in its local memory until this data is retrieved from a remote location. Remote Logging in essence is a bus monitor with filter rules on a KNX device. Any remote system retrieving and analyzing this captured data needs at least the semantic information about the group addresses including which devices are using these group addresses for which communication objects.

In a Remote Configuration situation a supervisory system operator may want to change a parameter on a KNX device like the time delay before OFF. This requires even more information about the KNX subnetwork and its devices.

4 The Solution

4.1 eXtensible Markup Language (XML)

As mentioned above, an electronically readable and manageable format for a common description of a KNX network has been missing.

A solution could be to declare a specific data base configuration and its binary format as the common description. This has draw-backs like confinement to a specific data base vendor or operating system. Additions or changes to the data base lead to possible incompatibilities between versions. This is undesired if the information required by e.g. a supervisory system has not changed.

In contrast, the eXtensible Markup Language (XML) has a number of advantages:

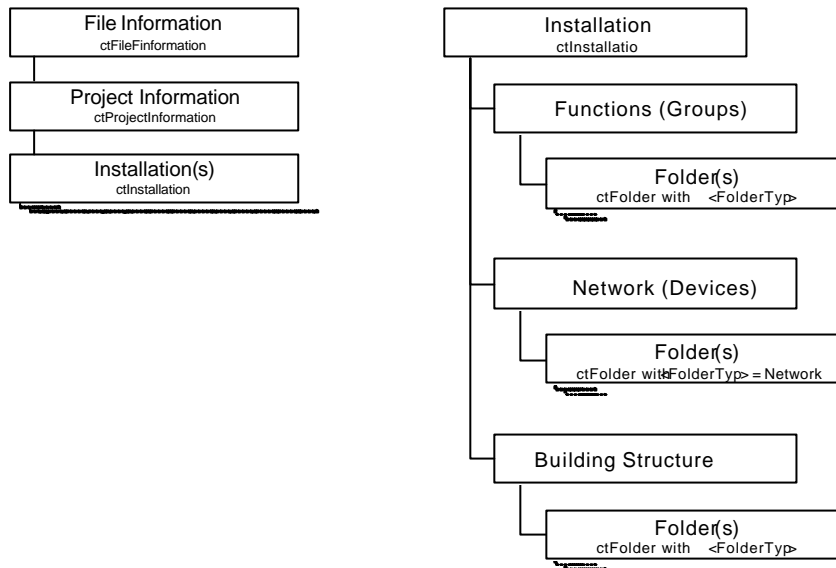
- (a) Commercial and Open Source Tools for manipulating XML files are available on all major operating system platforms;
- (b) Additions to the XML description can be made while maintaining backwards compatibility;
- (c) Tools for transforming the XML description to other formats and vice versa are readily available;
- (d) Using XML Schemas the XML export/import data can be tested for syntactical correctness;
- (e) The XML file can be read with a standard tool e.g. Internet Explorer.

The Task Force IP chose XML data encoding for the data exchange between ETS and other systems.

4.2 General structure of XML Data Encoding for KNX

The XML Data Encoding assumes that the data is organized in one or more files generated by a tool. A file contains information pertaining to a specific project with at least one installation and a maximum of 15 installations. This project information ensures that information organized across different files can be merged together into one file or data base again. Information on an installation is organized in a tree structure similar to a directory structure. There are three major branches: the Functional View (Group Addresses, Functions of Devices), the Building View (Physical Location of Devices), and the Network View (Communication Topology Location of Devices).

The XML file includes the above mentioned parts for the file information and project information. It then continues with a collection of installations that are part of this project. Currently the ETS3 supports only one installation, but the schema allows for having max 15 installations in one project.



The Installation as a root node includes the containers for the functions (group addresses structure), the network (topology) and building structure. Each is a collection of structural folders. These folders can be of a special type (e.g. location folders, network folders or function folders) or simply give a range of addresses a structural name. The traditional Main- and Subgroups structures are examples for such named collections of a given address range. Any folder –regardless of its type- can either carry multiple sub-folders of any type or -in the lowest structural level- a collection of e.g. group address or device elements.

4.3 File Information

The XML file information SHALL be a part of any XML file generated as an export from a configuration tool. XML files provided as templates for import into a configuration tool MAY contain this XML file information.

Table 1 lists all the XML elements, the XML data type of the data, a brief description and information about multiplicity (if mandatory or optional) for each XML element.

Element	Type	Description	Multiplicity
e:Date	date	The UTC date and time of generation of this XML file. Encoded according to the definition of the xsd:date datatype in ISO 8601 format.	1
e:Generator	string	Information about the software tool that generated this XML File. May be empty.	1
a:Name	string	Name of the software tool that created/ exported this XML file, e.g. ETS	1
a:Version	string	Version of the software tool that created/ exported this XML file, e.g. 3.0b	1
a:Owner	string	Name of the organization or user owning the software that generated this XML file.	1

Table 1 – Complex Type ctFileInformation

4.4 Project Information

The Project information SHALL be a part of any XML file generated as an export from a configuration tool. XML files provided as templates for import into a configuration tool MAY contain this Project information.

Table 2 lists all the XML elements, the XML data type of the data, a brief description and information about multiplicity (if mandatory or optional) for each XML element.

Element	Type	Description	Multiplicity
<i>a:Name</i>	string	User defined name of the project	1
<i>a:ID</i>	unsignedShort	12 Bit Project specific part of the Project-Installation ID (see EIBnet/IP Chapter2: Core)	1
<i>a:ContractNumber</i>	string	User defined contract number	1
<i>a>LastModified</i>	dateTime	Date and time of last project modification	1

Table 2 – Complex Type ctProjectInformation

4.5 Functional View

The Functional View organizes and presents Group Addresses in a generic structure that is independent of yet backwards compatible with the traditional 2- and 3-tiered group address scheme.

The group address structure defines how the 16 bit group addresses shall be interpreted. It uses a more flexible approach on sorting and categorizing the group addresses used in a KNX project than the ETS3 provides to the user.

Traditionally group addresses were structured into either 2 (Main/ Sub group address) or 3 (Main/ Middle/ Sub group address) parts. These two definitions assume a specific assignment of bits to represent the group address parts. For Main-Sub group address presentation of the group address the 16 bits are divided to 1:4:11, where the first bit is not used, the next four bits represent the Main group address and the remaining 11 bits constitute the Sub group address. For the 3-tiered group address the schema is 1:4:3:8 (Not_used:Main:Middle:Sub). This more flexible group address structure allows for including different structuring elements e.g. room or function information. It is possible to map this information either to numeric ranges (compare with traditional structure) or have this information additionally included to the traditional, numeric structure. This additional context information is irrelevant for the data transmitted across a KNX subnetwork but is important for the user interface.

If the XML data follows several restrictions and guidelines (templates will be provided) then it is possible for the ETS3 to convert the traditional group address structure from this extended structure and vice versa.

The group address list is a list of 16 bit group addresses. Each group address is defined by its unique address, a name, and a description.

Element	Type	Description	Multiplicity
e:Address	stKnxAddress	The group address number	1
a:Name	string	User defined name of the group address	1
a:Unfiltered	boolean	If true this element defines that the group address shall be forwarded unfiltered.	1
a:Central	boolean	If true this element defines that the group address represents a central function.	1
e:Description	string	A user defined description of the group address	0..1
e:DataLength	unsignedByte	Element with the data type length of the group address in Bits	1
a:DPTMainType	unsignedShort	Main data point type associated with this group address	0..1
a:DPTSubType	unsignedShort	Sub data point type associated with this group address	0..1
e:Association	sequence	This element contains a single assignment of a communication object with this group address	0..n
a:DeviceAddresses	stKnxAddress	This tag defines the individual address of the device containing the communication object assigned to this group address.	1
a:ObjectNumber	short	This tag defines the communication object number in a device assigned to this group address.	1

Table 3 – Complex Type ctGroupAddress

5 Outlook

The KNX Task Force IP has defined the general structure of the XML Data Encoding.

The approach for the Functional View (group address structure) is exemplary of the intention to not only export the current data base structure as-is but to define a schema that provides more information and venues for future enhancements to an engineering tool.

This takes careful consideration and balancing of future needs and backwards compatibility.

The details on the File Information, Project Information, and Functional View presented here are work-in-progress. Further work on details for information on Devices or on the Building Structure still needs to be done.

The result of this work will enable much simpler integration of KNX systems into other systems, reducing engineering effort and cost, which will in turn make KNX more attractive to the market.