# System Specifications

## KNXnet/IP

### Device Management

3

8

3

Summary

This document provides the KNXnet/IP Device Management specification.

Version 01.06.02 is a KNX Approved Standard.

This document is part of the KNX Specifications v2.1.

## Document updates

| Version | Date | Modifications |
|---|---|---|
| 1.0 DP | 2004.01.07 | Final version for Release for Voting |
| 1.1 DV | 2005.05.30 | Preparation of the Draft for Voting. |
| 1.2 DV | 2006.06.14 | Preparation of the Draft for Voting. |
| 1.3 AS | 2008.07.02 | Publication of the Approved Standard. |
| 1.3 AS | 2009.06.29 | Editorial update in view of inclusion in the KNX Specifications v2.0. |
| 1.4.00 AS | 2009.11.06 | • **AN118 "cEMI Transport Layer"** integrated. |
| 1.5.00 AS | 2010.07.26 | • **AN117 "KNX IP Communication Medium"** integrated |
| 01.06.00 | 2013.07.18 | • **AN139 "Procedures for the assignment of IAs to KNXnet/IP Tunnelling connections"** integrated. |
| 01.06.01 | 2013.07.18 | • Editorial review. |
| 01.06.02 | 2013.10.28 | Editorial updates for the publication of KNX Specifications 2.1. |

## References

A general reference is made to the RFCs [1] defining the Internet Protocol. These documents can be obtained on the Internet at http://www.ietf.org/rfc.html.

| | | |
|---|---|---|
| [01] | Chapter 3/5/1 | "Resources" |
| [02] | Chapter 3/6/3 | "External Message Interface" |
| [03] | Chapter 3/8/1 | "KNXnet/IP Overview" |
| [04] | Chapter 3/8/2 | "KNXnet/IP Core" |
| [05] | Chapter 3/8/4 | "KNXnet/IP Tunnelling" |
| [06] | Chapter 3/8/5 | "KNXnet/IP Routing" |
| [07] | Volume 6 | "Profiles" |

| | |
|---|---|
| Filename: | 03_08_03 Management v01.06.02 AS.docx |
| Version: | 01.06.02 |
| Status: | Approved Standard |
| Savedate: | 2013.10.28 |
| Number of pages: | 24 |

[1] Request for Comment: Internet Standards defined by the Internet Engineering Task Force (IETF) are firstly published as RFCs.

## Contents

# 1 Introduction

## 1.1 Scope

This specification defines the integration of KNX protocol implementations on top of Internet Protocol (IP) networks, called KNXnet/IP. It describes a standard protocol for KNX devices connected to an IP network, called KNXnet/IP devices. The IP network acts as a fast (compared to KNX transmission speed) backbone in KNX installations.

An overview of KNXnet/IP is presented in [03].

General frame descriptions and data exchange protocols between KNXnet/IP devices are specified in [04].

This Chapter 3/8/3 “Device Management” describes point-to-point exchange of IP datagrams over an IP network between a KNXnet/IP device acting as a server and a KNXnet/IP client for remote configuration and management of the KNXnet/IP device.

This document defines a standard protocol that is implemented within KNXnet/IP devices and the Engineering Tool Software (ETS®) to support KNX data exchange over non-KNX networks.

## 1.2 Definitions, acronyms and abbreviations

Refer to [03] for a list of definitions for the KNXnet/IP specification.

# 2 KNXnet/IP Device Management

## 2.1 Introduction

KNX devices connected to an IP network through KNXnet/IP devices shall be configured by ETS as described in [05].

KNXnet/IP Device Management defines management of KNXnet/IP devices.

IP unicast addressing shall be used for direct communication with individual KNXnet/IP devices.

## 2.2 General remarks

As a KNXnet/IP device is connected to two different networks, two different ways of configuration are imaginable: via KNX and via IP. Every KNXnet/IP device should allow configuration by means of the KNXnet/IP device configuration protocol implementation (IP). Implementation of configuration using KNX is mandatory if the KNXnet/IP device physically connects to a KNX Subnetwork.

For download, all parameter values shall be presented in “big endian” format (Motorola-like), so the most significant octet shall always be transferred firstly.

## 2.3 Configuration and Management

### 2.3.1 Introduction

A KNXnet/IP device shall implement configuration and management using KNX Interface Objects for application and parameter download. Basically, KNXnet/IP device configuration and management using IP shall be based on Interface Object Properties and shall follow the same pattern as using KNX. The KNXnet/IP protocol defined for this purpose has the advantage of supporting large data structures.

Any KNXnet/IP device shall accept configuration through KNX Network telegrams. A KNXnet/IP Router shall accept configuration through KNXnet/IP Routing frames if those frames carry cEMI frames addressed to the KNX Individual Address of the KNXnet/IP Router.

### 2.3.2 KNXnet/IP endpoints and service containers

A KNXnet/IP Server shall implement one service container for each KNX Subnetwork connected to the server. While it shall implement at least one service container, it may implement more than one. If the KNXnet/IP Server supports more than one KNX Subnetwork connection, it is required that every KNX Subnetwork is represented by a different control endpoint.

A KNXnet/IP client shall therefore consider every service container represented by a control endpoint as one independent entity no matter if they are implemented in only one or two separate KNXnet/IP Servers. In addition, multiple service containers in a single KNXnet/IP Server shall behave exactly like multiple service containers in multiple KNXnet/IP Servers (e.g. regarding IP traffic).

The control endpoints exposed by KNXnet/IP Servers can be discovered according to the discovery procedure described in [04].

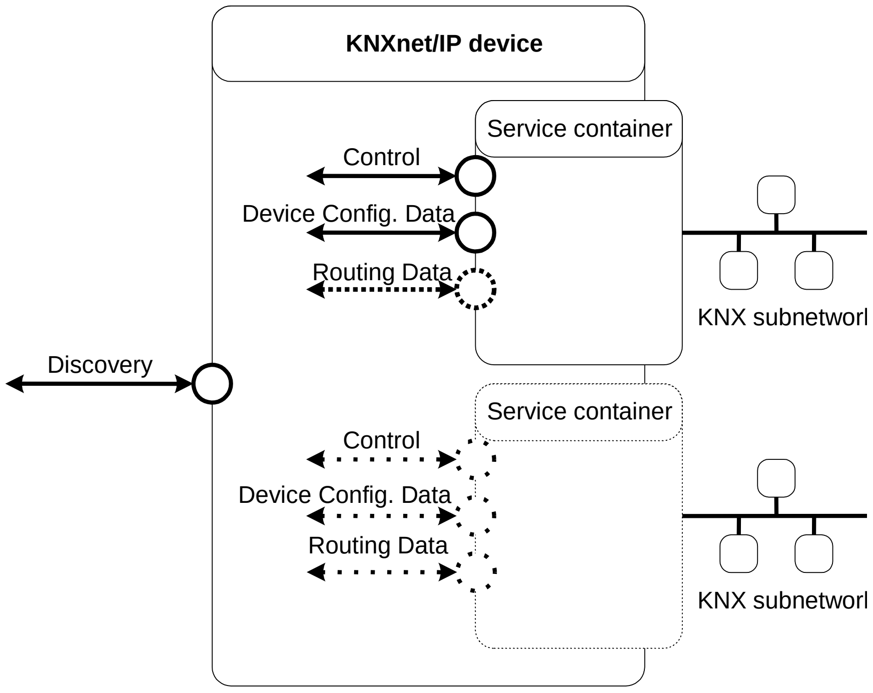

**Figure 1 – KNXnet/IP device endpoints**

A KNXnet/IP device's service container shall implement a Device Configuration and Management end point. Further service end points like Routing Data may be implemented depending on the device function.

The configuration and management end point shall allow access to device parameters, Filter Table etc after a connection is established through the control endpoint. It shall be used by the device to exchange configuration information with ETS. Every KNXnet/IP device shall use the "KNXnet/IP Device HPAI" (unicast IP address and port number) for the Device Configuration and management data endpoint.

To access configuration and management, a KNXnet/IP client shall connect to the respective data endpoint according to [04]. All communication traffic following that connection shall be handled by DEVICE_CONFIGURATION_REQUEST - and DEVICE_CONFIGURATION_ACK frames.

If the KNXnet/IP client does not receive a DEVICE_CONFIGURATION_ACK within the DEVICE_CONFIGURATION_REQUEST_TIMEOUT (= 10 seconds) or the status of a received DEVICE_CONFIGURATION_ACK frame signals any kind of error condition, the client shall repeat the DEVICE_CONFIGURATION_REQUEST three (3) times and then terminate the connection by sending a DISCONNECT_REQUEST to the server's control endpoint.

The KNXnet/IP Server shall send no DEVICE_CONFIGURATION_ACK and shall discard the frame if it receives a frame with an unexpected sequence number.

## 2.4 Interface Object Types

KNXnet/IP and KNX IP devices shall be managed mainly through Properties of Interface Objects. For the KNXnet/IP Device Management specific functionality, the following Interface Object Types shall mainly be used:

- the Device Object, and
- the KNXnet/IP Parameter Object

For the specification of which Interface Object Types are mandatory or optional for a KNXnet/IP – or KNX IP device, please refer to [07].

For the specification of the Properties of the Device Object, please refer to [01].

## 2.5 KNXnet/IP Parameter Object

### 2.5.1 Overview

The KNXnet/IP Parameter Object shall include the IP parameters for the KNXnet/IP device's service container.

Several Properties of the KNXnet/IP Parameter Object are closely related to each other.

EXAMPLE 1 PID_IP_ADDRESS and PID_SUBNET_MASK are related to each other.

Therefore write accesses to the Properties shall always use KNX Transport Layer connections.

Following is a specification of Properties of the KNXnet/IP Parameter Object.

### 2.5.2 PID_PROJECT_INSTALLATION_ID (PID = 51)

- Property name: Project Installation Identification
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

This Property shall be set by ETS only and expresses which project and installation this KNXnet/IP device is part of.

The 16 bit value shall contain the project number in the upper 12 bits and the installation number in the lower 4 bits. The factory default value for this property shall be set to 0000h.

| octet 2 | | | | | | | | octet 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| project number | | | | | | | | | | | | installation number | | | |

**Figure 2 – PID_PROJECT_INSTALLATION_ID**

ETS shall assign the proper value based on the ETS project number and the number of installations in a project.

The project number shall always be in the range [1 … 4095].

If a project has only one installation, the installation number of that installation shall be set to zero (0).

If a project has more than one installation then the installation number shall be set to the number of the installation in the range [1 … 15].

### 2.5.3 PID_KNX_INDIVIDUAL_ADDRESS (PID = 52)

- Property name: KNX Individual Address
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

This Property shall be implemented by any KNXnet/IP device.

This Property shall contain the KNX Individual Address of the KNXnet/IP device. The device shall ensure that the value of this Property is in sync with the combined value of the corresponding Device Object properties.

### 2.5.4 PID_ADDITIONAL_INDIVIDUAL_ADDRESSES (PID = 53)

- Property name: Additional Individual Addresses
- Property Datatype: PDT_UNSIGNED_INT[]
- Datapoint Type: None.

This Property shall be implemented by devices providing KNXnet/IP Routing or KNXnet/IP Tunnelling.

This Property shall contain a list of additional KNX Individual Addresses. The first entry in the list shall be the length of the list. Subsequent entries shall be KNX Individual Addresses.

**Usage by the Management Server (device)**

The Management Server shall use the entries in PID_¬ADDITIONAL_¬INDIVIDUAL_-ADDRESSES as additional Individual Addresses for KNXnet/IP Tunnelling Connections.

If a KNXnet/IP Tunnelling connection is opened, then the Management Server shall use the first free Additional Individual Address in this list. This is; the Management Server shall use the IA at the lowest Property array element index for which no KNXnet/IP Tunnelling exist already.

- If no free IA is found in this list and all IA entries in the list are unique, then the Management Server shall according [05] respond to the KNXnet/IP Tunnelling Client that no more KNXnet/IP Tunnelling connections can be established unless the own IA (PID_INDI¬VI¬DUAL_ADDRESS) can also be used for KNXnet/IP Tunnelling and is not in use for any KNXnet/IP Tunnelling connection.
- If the KNXnet/IP Tunnelling server finds one or more free entries in PID_¬ADDITIONAL_-INDIVIDUAL_¬ADDRESSES but one or more of those free entries
  - have the same IA as used for an open KNXnet/IP Tunnelling connection [2)], or
  - have the same value as the IA own IA of the device and the device does not use its own IA for KNXnet/IP Tunnelling,
- then the Management Server shall according [05] clause 2.2.2 “Tunnelling on KNX Data Link Layer” respond to the KNXnet/IP Tunnelling Client that no more KNXnet/IP Tunnelling connections can be established with a unique Individual Address, this is, the CONNECT_-RESPONSE shall contain the error code E_NO_MORE_¬UNIQUE_¬CONNECTIONS.

These entries shall apply immediately. If any entry is changed by a Management Client in any way, it shall apply immediately, without restart, reset or closing/re-opening a possible open KNXnet/IP Tunnelling Connection.

### 2.5.5 PID_CURRENT_IP_ASSIGNMENT_METHOD (PID = 54)

- Property name: Current IP Assignment Method
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall capture the IP address assignment method employed to set the current IP address.

The IP assignment methods shall be identified as follows:

| Value of PID_CURRENT_IP_ ASSIGNMENT_METHOD | Assignment method |
|---|---|
| 1 | manually |
| 2 | BootP |
| 4 | DHCP |
| 8 | AutoIP |

Only one value shall be set at a time.

2) This can only be the case if two or more entries in PID_ADDITIONAL_INDIVIDUAL_ADDRESSES are filled with the same IA-value or the same value as the own IA of the KNXnet/IP Tunnelling Server device.

### 2.5.6 PID_IP_ASSIGNMENT_METHOD (PID = 55)

- Property name: IP Assignment Method
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall show the enabled IP address assignment methods for setting the current IP address. At least one shall be enabled.

The supported assignment methods are defined as follows.

| Value of PID_CURRENT_IP_ ASSIGNMENT_METHOD | Assignment method |
|---|---|
| 1 | manually |
| 2 | BootP |
| 4 | DHCP |
| 8 | AutoIP |

This Property shall determine the behaviour of the IP address assignment procedure as specified in [04].

### 2.5.7 PID_IP_CAPABILITIES (PID = 56)

- Property name: IP Capabilities
- Property Datatype: PDT_BITSET8
- Datapoint Type: None.

This Property shall show the IP capabilities supported by the KNXnet/IP device.

Property bits 0, 1, and 2 shall show the available IP address assignment methods for setting the current IP address. Manual address assignment shall always be available.

**Table 1 – Device Capabilities**

| Bit | Description |
|---|---|
| 0 | BootP |
| 1 | DHCP |
| 2 | AutoIP |
| 3 | Reserved |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

This Property is optional but its information shall be available in the ETS database.

### 2.5.8 PID_CURRENT_IP_ADDRESS (PID = 57)

- Property name: Current IP Address
- Property Datatype: PDT_UNSIGNED_LONG
- Datapoint Type: None.

This Property shall contain the currently used IP v4 address (32 bit).

This value shall be read-only and shall be set as specified in [04] clause “IP Address Assignment”.

### 2.5.9 PID_CURRENT_SUBNET_MASK (PID = 58)

| | |
|---|---|
| • Property name: | Current Subnet Mask |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall contain the currently used IP subnet mask (32 bit).

### 2.5.10 PID_CURRENT_DEFAULT_GATEWAY (PID = 59)

| | |
|---|---|
| • Property name: | Current Default Gateway |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall contain the IP address of the currently used default gateway.

### 2.5.11 PID_IP_ADDRESS (PID = 60)

| | |
|---|---|
| • Property name: | IP Address |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall contain the configured fixed IP v4 address (32 bit) value. This Property shall contain the IP address as configured by a configuration tool. It shall be used if a manual address assignment is enabled. IP Address Assignment is described in [04] clause “IP Address Assignment”.

### 2.5.12 PID_SUBNET_MASK (PID = 61)

| | |
|---|---|
| • Property name: | Subnet Mask |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall contain the configured IP subnet mask (32 bit). This Property shall contain the IP subnet mask as configured by a configuration tool. It shall be used when a manual address assignment is enabled. IP Address Assignment is described in [04] clause “IP Address Assignment”.

### 2.5.13 PID_DEFAULT_GATEWAY (PID = 62)

| | |
|---|---|
| • Property name: | Default Gateway |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall contain the configured IP address of the default gateway. This Property shall contain the IP address of the default gateway as configured by a configuration tool. It shall be used when a manual address assignment is enabled. IP Address Assignment is described in [04] clause “IP Address Assignment”

### 2.5.14 PID_DHCP_BOOTP_SERVER (PID = 63)

| | |
|---|---|
| • Property name: | DHCP/BootP Server |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall contain the IP address of the DHCP/BootP server the KNXnet/IP device last received its IP address from.

This Property shall be read-only.

### 2.5.15 PID_MAC_ADDRESS (PID = 64)

- Property name: MAC Address
- Property Datatype: PDT_GENERIC_06
- Datapoint Type: None.

This Property shall contain the MAC address (48 bit) of the KNXnet/IP device.

The value shall be set by the manufacturer and shall be read-only.

### 2.5.16 PID_SYSTEM_SETUP_MULTICAST_ADDRESS (PID = 65)

- Property name: System Setup Multicast Address
- Property Datatype: PDT_UNSIGNED_LONG
- Datapoint Type: None.

This Property shall contain the KNXnet/IP System Setup Multicast Address.

The value of this Property shall be fixed at 224.0.23.12.

### 2.5.17 PID_ROUTING_MULTICAST_ADDRESS (PID = 66)

- Property name: Routing Multicast Address
- Property Datatype: PDT_UNSIGNED_LONG
- Datapoint Type: None.

This Property shall contain the KNXnet/IP Routing Multicast Address. The default value of this Property shall be equal to the KNXnet/IP System Setup Multicast Address. The KNXnet/IP Routing Multicast Address may be set at run-time. A changed value shall become active after a reset of the KNXnet/IP device.

### 2.5.18 PID_TTL (PID = 67)

- Property name: Time To Live
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall hold the Time-To-Live (TTL) value for IP communication across IP network Routers. The default value shall be set to 16. This value may be changed.

### 2.5.19 PID_KNXNETIP_DEVICE_CAPABILITIES (PID = 68)

- Property name: KNXnet/IP Device Capabilities
- Property Datatype: PDT_BITSET16
- Datapoint Type: None.

The list of KNXnet/IP device capabilities shall provide information about which features are implemented in the KNXnet/IP device. A value of ‘1’ shall always mean “available”, ‘0’ shall mean “not available/not supported”.

**Table 2 – Device Capabilities**

| Bit | Description |
|---|---|
| 0 | Device Management |
| 1 | Tunnelling |
| 2 | Routing |
| 3 | Remote Logging |
| 4 | Remote Configuration and Diagnosis |
| 5 | Object Server |
| 6 | Reserved |
| 7 | Reserved |
| 8 | Reserved |
| 9 | Reserved |
| 10 | Reserved |
| 11 | Reserved |
| 12 | Reserved |
| 13 | Reserved |
| 14 | Reserved |
| 15 | Reserved |

### 2.5.20 PID_KNXNETIP_DEVICE_STATE (PID = 69)

- Property name: KNXnet/IP Device State
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall be implemented by any KNXnet/IP Server.

This Property shall contain status information of the KNXnet/IP device.

The Property shall be evented i.e. if the value of the Property changes the current value shall be sent using M_PropInfo.ind.

**Table 3 – Device State**

| Bit | Description |
|---|---|
| 0 | KNX Fault: is set if KNX network cannot be accessed |
| 1 | IP Fault: is set if IP network cannot be accessed |
| 2 | Reserved |
| 3 | Reserved |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

### 2.5.21 PID_KNXNETIP_ROUTING_CAPABILITIES (PID = 70)

- Property name: KNXnet/IP Routing Capabilities
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall be implemented by devices providing KNXnet/IP Routing.

This Property shall contain the description of the device’s KNXnet/IP Routing capabilities. It shall provide information about which features are implemented in the KNXnet/IP device.

A value of ‘1’ shall always mean “available” or “yes”, ‘0’ shall mean “not available/not supported” or “no”.

**Table 4 – Routing Capabilities**

| Bit | Description |
|---|---|
| 0 | Statistics, Queue overflow error count, implemented |
| 1 | Statistics, Transmitted telegram info count, implemented |
| 2 | Priority/FIFO implemented |
| 3 | Multiple KNX installations supported |
| 4 | Group Address mapping supported |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

If statistics for queue overflow error count (bit 0) or for transmitted telegram count (bit 1) is implemented then the corresponding bit is set. These are optional features.

If priority FIFO is implemented then the corresponding bit is set. This is an optional feature.

If multiple KNX installations are supported by a KNXnet/IP Router the corresponding bit is set. This is an optional feature.

If a KNXnet/IP Router supports mapping of Group Addresses between KNX installations the corresponding bit is set. This is an optional feature.

### 2.5.22 PID_PRIORITY_FIFO_ENABLED (PID = 71)

- Property name: Priority FIFO Enabled
- Property Datatype: PDT_BINARY_INFORMATION
- Datapoint Type: None.

This Property shall be implemented by devices implementing priority FIFO as described in [06] clause “Forwarding rules”.

This Property shall show if priority FIFO is enabled (1) or disabled (0).

### 2.5.23 PID_QUEUE_OVERFLOW_TO_IP (PID = 72)

- Property name: Queue Overflow to IP
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

This Property shall be implemented by devices providing KNXnet/IP Routing.

This Property shall contain the number of telegrams lost due to an overflow of the queue to the IP network.

This Property shall be an unsigned integer (~ 64 000) and is for informational purposes only. It may be reset by a scheme at the discretion of the manufacturer. The count does not wrap around when the maximum is reached.

### 2.5.24 PID_QUEUE_OVERFLOW_TO_KNX (PID = 73)

|  |  |
|---|---|
| • Property name: | Queue Overflow to KNX |
| • Property Datatype: | PDT_UNSIGNED_INT |
| • Datapoint Type: | None. |

This property shall be implemented by devices providing KNXnet/IP Routing.

This Property shall contain the number of telegrams lost due to an overflow of the queue to the KNX Subnetwork.

This Property shall be an unsigned integer (~ 64 000) and is for informational purposes only. It may be reset by a scheme at the discretion of the manufacturer. The count does not wrap around when the maximum is reached.

### 2.5.25 PID_MSG_TRANSMIT_TO_IP (PID = 74)

|  |  |
|---|---|
| • Property name: | Telegrams Transmitted to IP |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall be implemented by devices providing KNXnet/IP Routing.

This Property shall show the number of telegrams successfully transmitted to the IP network.

This shall include the number of KNXnet/IP telegrams of any kind (KNXnet/IP Core, KNXnet/IP Tunnelling, KNXnet/IP Routing, KNXnet/IP Device Management etc.) associated with any KNX Individual Address that is successfully sent on the IP network. This shall include KNXnet/IP ACK telegrams.

### 2.5.26 PID_MSG_TRANSMIT_TO_KNX (PID = 75)

|  |  |
|---|---|
| • Property name: | Telegrams Transmitted to KNX |
| • Property Datatype: | PDT_UNSIGNED_LONG |
| • Datapoint Type: | None. |

This Property shall be implemented by devices providing KNXnet/IP Routing.

This Property shall contain the number of telegrams successfully transmitted by a KNXnet/IP Router to the KNX Subnetwork.

For a KNX IP device this Property shall contain the number of telegrams received and forwarded to its application.

### 2.5.27 PID_FRIENDLY_NAME (PID = 76)

|  |  |
|---|---|
| • Property name: | Friendly Name |
| • Property Datatype: | PDT_UNSIGNED_CHAR[30] |
| • Datapoint Type: | None. |

This Property shall be implemented by any KNXnet/IP device.

This Property shall contain a string with a human readable (friendly) name for the KNXnet/IP device.

The device friendly name may be any NULL (00h) terminated ISO 8859-1 character string with a maximum length of 30 octets. Unused octets shall be filled with the NULL (00h) character.

### 2.5.28 PID_ROUTING_BUSY_WAIT_TIME (PID = 78)

| | |
|---|---|
| • Property name: | Routing Busy Wait Time |
| • Property Datatype: | PDT_UNSIGNED_INT |
| • Datapoint Type: | None. |

The Property PID_ROUTING_BUSY_WAIT_TIME SHALL be accessible via the KNXnet/IP Parameter Object of a KNXnet/IP Router or KNX IP device.

This Property shall hold the value for the wait time $t_w$ sent with a ROUTING_BUSY frame. The default value shall be 100 ms. The permissible value range is any integer value between 20 ms and 100 ms.

This Property is mandatory for devices implementing KNXnet/IP or KNX IP.

## 2.6 Transport Layer interface

### 2.6.1 Switching to the cEMI Transport Layer in the cEMI Server by opening a KNXnet/IP Device Management connection

#### 2.6.1.1 Use

This shall be implemented by devices that use the KNXnet/IP Device Management.

#### 2.6.1.2 Activation of the cEMI Transport Layer interface

If a KNXnet/IP Device Management connection is established, the Management Server shall implicitly switch its Transport Layer to “cEMI Transport Layer mode”. The cEMI Server Transport Layer shall issue a T_Connect.ind primitive to the remote Application Layer to indicate that a Transport Layer connection is established.

This Transport Layer operation mode shall last for the duration of the KNXnet/IP Device Management connection.

The Transport Layer operation mode shall be reset to its default value if the KNXnet/IP Device Management connection is broken by the KNXnet/IP Device Management Client or – Server (device, e.g. when the KNXnet/IP Device Management connection times out).

**General exception handling**

1. If a cEMI Server does not support the cEMI Transport Layer communication mode and it receives a cEMI T_Data_Individual.req – or cEMI T_Data_Connected.req frame, then
   - this frame shall be ignored by the cEMI Server, and
   - on IP, the originating KNXnet/IP DEVICE_CONFIGURATION_REQUEST frame that transports this cEMI-frame shall be confirmed by a KNXnet/IP DEVICE_CONFIGUARATION_ACK frame.

#### 2.6.1.3 Supervising connection timeouts on the host protocol

The KNXnet/IP or KNX IP device shall deactivate the cEMI Transport Layer Mode and return to the normal Transport Layer mode if the KNXnet/IP Device Management connection is broken.

NOTE 1 The KNXnet/IP Device Management may be broken by the KNXnet/IP Device Management Client (ETS), or autonomously by the KNXnet/IP Device Management Server (device) if the KNXnet/IP Device Management connection times out.

#### 2.6.1.4 T_Data_Individual.con and T_Data_Connected.con

The cEMI Transport Layer instance shall provide the T_Data_Individual.con respectively the T_Data_Connected.con to the Application Layer after reception of the DEVICE_CONFIGURATION_ACK frame.

#### 2.6.1.5 Controlling sequencing of the cEMI messages

This is guaranteed by the KNXnet/IP Device Management protocol.

#### 2.6.1.6 Deactivation of the cEMI Transport Layer interface

The cEMI Server shall deactivate the cEMI Transport Layer interface if the KNXnet/IP Device Management connection is closed.

This may be caused by any of the following (not exclusive)

- a request by the KNXnet/IP Device Management Client to close the KNXnet/IP Device Management connection, or
- a time-out of the KNXnet/IP connection,
- etc.

When the cEMI Server Transport Layer is deactivated, the cEMI Server shall issue a T_Disconnect.ind primitive to the remote Application Layer to indicate that the Transport Layer connection is closed.

# 3 Implementation rules and guidelines

## 3.1 Introduction

This clause of the specification describes the implementation details and features of KNXnet/IP Device Management.

## 3.2 Discovery and self description

Every KNXnet/IP device shall support discovery and self description according to [04].

## 3.3 Device Management

A KNXnet/IP device should implement Device Management (see [03] clause “KNXnet/IP device classes”).

Device Management shall use the common KNXnet/IP port number 3671.

## 3.4 Security

Device Management frames are not encrypted or otherwise secured.

## 3.5 Error handling

### 3.5.1 Introduction

Some errors may occur in normal operation, e.g. due to unplugged network connections. These errors are reflected in the device parameter object.

If a Property access failure occurs, this is noted in the configuration response.

### 3.5.2 KNX net failure

Should the KNXnet/IP device not be able to transmit telegrams over the KNX Subnetwork for five seconds, the corresponding bit in the PID_KNXNETIP_DEVICE_STATE Property shall be set to ‘1’. If communication can be resumed, the bit shall be set to ‘0’.

### 3.5.3 IP net failure

Should communication to the IP network fail for more than five seconds the corresponding bit in the PID_KNXNETIP_DEVICE_STATE Property shall be set to ‘1’. If communication can be resumed, the bit shall be set to ‘0’.

### 3.5.4 Queue overflow

Should one of the two routing queues overflow and queue overflow statistics are implemented (see 2.5.21), the corresponding counter is increased.

## 3.6 Device statistics and status information

A KNXnet/IP device shall provide statistics and status information, see also 2.5 (KNXnet/IP Parameter Object specification). All values are unsigned, and all counts do not wrap around when the maximum value is reached (Property can then be set to 0).

**Table 5 – Device statistics**

| Name | Type | Size | Description |
|---|---|---|---|
| Queue overflow to IP | Error | 2 octets | Number of telegrams lost because queue to IP overflowed |
| Queue overflow to KNX | Error | 2 octets | Number of telegrams lost because queue to KNX overflowed |
| Telegrams transmitted to IP | Info | 4 octets | Number of telegrams successfully transmitted to IP |
| Telegrams transmitted to KNX | Info | 4 octets | Number of telegrams successfully transmitted to KNX |

# 4 Frame formats

## 4.1 Introduction

All KNXnet/IP frames shall have a common header, consisting of the protocol version, length information, and the KNXnet/IP service type identifier.

Requests sent to connectionless KNXnet/IP endpoints shall include information about the return address. This HPAI for the reception of the response information shall always be the first data of the KNXnet/IP body of all these requests. Additional, KNXnet/IP service related data may follow. Response frames shall not contain this kind of return address information.

## 4.2 IP Configuration and Management

### 4.2.1 Device management services

Two service types are defined for Device Management of KNXnet/IP devices: DEVICE_CONFIGURATION_REQUEST and DEVICE_CONFIGURATION_ACK.

**Table 6 – KNXnet/IP Device Management service type identifiers**

| Service name | Code | V. | Description |
|---|---|---|---|
| DEVICE_CONFIGURATION_REQUEST | 0310h | 1 | Reads/Writes KNXnet/IP device configuration data (Interface Object Properties). |
| DEVICE_CONFIGURATION_ACK | 0311h | 1 | Sent by a KNXnet/IP device to confirm the reception of the DEVICE_CONFIGURATION_REQUEST. |

### 4.2.2 Connection Type

The connection type value for the connection type DEVICE MGMT_CONNECTION shall be 03h.

Please refer to [04] for more details.

### 4.2.3 Connection Request Information

The Connection Request Information (CRI) shall have no options.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   DEVICE_MGMT_CONNECTION      |
|   (1 Octet = 02h)             |   (1 Octet = 03h)             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

**Figure 3 – KNXnet/IP Device Management CRI binary format**

### 4.2.4 Connection Response Data Block

The Connection Response Data Block (CRD) shall have no options.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   DEVICE_MGMT_CONNECTION      |
|   (1 Octet = 02h)             |   (1 Octet = 03h)             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

**Figure 4 – KNXnet/IP Device Management CRD binary format**

### 4.2.5 Configuration Message

DEVICE_CONFIGURATION_REQUEST shall carry a cEMI frame with the configuration message. The configuration message shall contain a local device management service as defined by [02]

| | **KNXnet/IP Device Management version** | |
|---|---|---|
| **Supported services** | **1** | **2** |
| KNXnet/IP Client → KNXnet/IP Server | ▨ | ▨ |
| - M_PropRead.req | M | M |
| - M_PropWrite.req | M | M |
| - M_Reset.req | M | M |
| - M_FuncPropCommand.req | M | M |
| - M_FuncPropStateRead.req | M | M |
| - cEMI T_Data_Individual.req | X | M |
| - cEMI T_Data_Connected.req | X | M |
| KNXnet/IP Server → KNXnet/IP Client | ▨ | ▨ |
| - M_PropRead.con | M | M |
| - M_PropWrite.con | M | M |
| - M_PropInfo.ind | M | M |
| - M_FuncPropStateResponse.con | M | M |
| - cEMI T_Data_Individual.ind | X | M |
| - cEMI T_Data_Connected.ind | X | M |

These Property services shall use Interface Object Type and Interface Object Instance instead of local Interface Object Index as addressing scheme for the device management. M_PropInfo.ind shall be used for the evented DeviceState Property.

M_FuncPropCommand.req, M_FuncPropStateRead.req, and M_FuncPropStateRead.con are specified in [02].

### 4.2.6 DEVICE_CONFIGURATION_REQUEST

```
                         KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DEVICE_CONFIGURATION_REQUEST                                |
|   (0310h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(Connection Header) +                |
|                                           sizeof(cEMI frame) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                          KNXnet/IP body
                         Connection header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Communication Channel ID    |
|   (1 Octet)                   |   (1 Octet)                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Sequence Counter            |   reserved                    |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
                            cEMI frame
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Message Code                |   Service Information         |
|   (1 Octet)                   |   (variable length)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

**Figure 5 – DEVICE_CONFIGURATION_REQUEST frame binary format**

### 4.2.7 DEVICE_CONFIGURATION_ACK

The configuration response includes the configuration header as well as a status field. The status field indicates success or failure of the requested operation, providing some error information.

```
                         KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DEVICE_CONFIGURATION_ACK                                    |
|   (0311h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(Connection Header)                  |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                          KNXnet/IP body
                         Connection header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Communication Channel ID    |
|   (1 Octet)                   |   (1 Octet)                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Sequence Counter            |   Status                      |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
```

**Figure 6 - DEVICE_CONFIGURATION_ACK frame binary format**

**Table 7 – Configuration status codes**

| Constant Name | Value | Description |
|---|---|---|
| E_NO_ERROR | 00h | The message was received successfully. |

# 5 Binary examples of KNXnet/IP frames

## 5.1 DEVICE_CONFIGURATION_REQUEST

```
        +-------------------------------+ - - - - KNXnet/IP header - - - -
  1     |              06h              |    header size
        +-------------------------------+
  2     |              10h              |    protocol version
        +-------------------------------+
  3     |              03h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0310h
  4     |              10h              | /
        +-------------------------------+
  5     |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, nn octets
  6     |              nnh              | /
        +-------------------------------+ - - - - connection header - - - -
  7     |              04h              |    structure length of connection header
        +-------------------------------+
  8     |              15h              |    communication channel ID, e.g. 21
        +-------------------------------+
  9     |              00h              |    sequence counter
        +-------------------------------+
 10     |              00h              |    reserved
        +-------------------------------+ - - - - cEMI frame - - - -
 11     |              FCh              |    Message Code, e.g. M_PropRead.req
        +-------------------------------+
 12     |              ...              | \
        +- - - - - - - - - - - - - - - -+ |
 ..     |              ...              |  > Service Information
        +- - - - - - - - - - - - - - - -+ |
 nn     |              ...              | /
        +-------------------------------+
```

**Figure 7 – DEVICE_CONFIGURATION_REQUEST frame binary format: example**

## 5.2 DEVICE_CONFIGURATION_ACK

```
        +-------------------------------+ - - - - KNXnet/IP header - - - -
  1     |              06h              |    header size
        +-------------------------------+
  2     |              10h              |    protocol version
        +-------------------------------+
  3     |              03h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0311h
  4     |              11h              | /
        +-------------------------------+
  5     |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, 10 octets
  6     |              0ah              | /
        +-------------------------------+ - - - - connection header - - - -
  7     |              04h              |    structure length of connection header
        +-------------------------------+
  8     |              15h              |    communication channel ID, e.g. 21
        +-------------------------------+
  9     |              00h              |    sequence counter
        +-------------------------------+
 10     |              00h              |    status
        +-------------------------------+
```

**Figure 8 – DEVICE_CONFIGURATION_ACK frame binary format: example**

# 6 Certification

## 6.1 Introduction

This clause provides information on the test procedures and requirements of the certification process.

## 6.2 Service support matrix

| Service name | sent from ... to ... | implementation is |
|---|---|---|
| DEVICE_CONFIGURATION_REQUEST | Client → Server<br>Server → Client | M |
| DEVICE_CONFIGURATION_ACK | Client → Server<br>Server → Client | M |

Legend: “M” = Mandatory, “O” = Optional, “n.a.” = not applicable

## 6.3 Test cases

### 6.3.1 Introduction

Listed here are a number of test cases a KNXnet/IP Router implementation should be checked against.

### 6.3.2 Normal operation

Normal operation means device configuration.

| Nr. | Description | Expected Result | |
|---|---|---|---|
| 1.1 | Device receives configuration telegram from KNX subnet addressed to the device. | If valid configuration telegram | parameters are set. |
| | | If invalid configuration telegram is received | error message is returned. |
| 1.2 | Device receives configuration telegram from IP. | If valid configuration telegram | parameters are set. |
| | | If invalid configuration telegram is received | error message is returned. |

### 6.3.3 Parameterisation through IP

This requires a valid connection through the control endpoint of the service container, which in turn has to be searched by the discovery service before.

| Nr. | Description | Expected Result |
|---|---|---|
| 2.1 | Read device capabilities via a DEVICE_CONFIGURATION_REQUEST | A DEVICE_CONFIGURATION_ACK message with configuration status E_NO_ERROR and 4 bytes of data. |