

BS EN 13321-2:2012



BSI Standards Publication

Open Data Communication in Building Automation, Controls and Building Management — Home and Building Electronic Systems

Part 2: KNXnet/IP Communication

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of EN 13321-2:2012. It supersedes BS EN 13321-2:2006 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee RHE/16, Performance requirements for control systems.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2013.

Published by BSI Standards Limited 2013

ISBN 978 0 580 77057 9

ICS 35.240.99; 97.120

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2013.

Amendments issued since publication

Date	Text affected
------	---------------

EUROPEAN STANDARD
NORME EUROPÉENNE
EUROPÄISCHE NORM

EN 13321-2

December 2012

ICS 35.240.99; 97.120

Supersedes EN 13321-2:2006

English Version

**Open Data Communication in Building Automation, Controls and
Building Management - Home and Building Electronic Systems -
Part 2: KNXnet/IP Communication**

Réseau ouvert de communication de données pour
l'automatisation, la régulation et la gestion technique du
bâtiment - Systèmes électroniques pour la maison et le
bâtiment - Partie 2: Communication KNXnet/IP

Offene Datenkommunikation für die Gebäudeautomation
und Gebäudemanagement - Elektrische Systemtechnik für
Heim und Gebäude - Teil 2: KNXnet/IP-Kommunikation

This European Standard was approved by CEN on 30 September 2012.

CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

© 2012 CEN All rights of exploitation in any form and by any means reserved
worldwide for CEN national Members.

Ref. No. EN 13321-2:2012: E

Contents

Page

Foreword.....	4
Introduction	5
1 Scope	7
2 Normative references	8
3 Terms and definitions	8
4 Symbols and abbreviations	10
5 Requirements	11
5.1 Clause 1: Overview	11
5.1.1 KNXnet/IP Document Clauses	11
5.1.2 Mandatory and optional implementation of IP protocols	13
5.1.3 Security considerations	15
5.2 Clause 2: Core	17
5.2.1 Scope	17
5.2.2 KNXnet/IP frames	17
5.2.3 Host protocol independence	19
5.2.4 Discovery and self description.....	20
5.2.5 Communication Channels	21
5.2.6 General implementation guidelines	24
5.2.7 Data Packet structures	28
5.2.8 IP Networks	42
5.2.9 Certification	47
5.3 Clause 3: Device Management Specification.....	48
5.3.1 Scope	48
5.3.2 KNXnet/IP Device Management.....	48
5.3.3 Implementation rules and guidelines	60
5.3.4 Data packet structures	62
5.3.5 Certification	65
5.4 Clause 4: Tunnelling.....	66
5.4.1 Scope	66
5.4.2 Tunnelling of KNX telegrams.....	66
5.4.3 Configuration and Management.....	70
5.4.4 Frame structures.....	70
5.4.5 Certification	72
5.5 Clause 5: Routing	73
5.5.1 Scope	73
5.5.2 KNXnet/IP Routing of KNX telegrams.....	73
5.5.3 Implementation rules and guidelines	80
5.5.4 Configuration and Management.....	83
5.5.5 Data packet structures	83
5.5.6 Certification	85
5.6 Clause 6: Remote Diagnosis and Configuration	86
5.6.1 Scope	86
5.6.2 Remote Diagnosis of KNXnet/IP devices	87
5.6.3 Configuration and Management.....	87
5.6.4 Data packet structures	88
5.6.5 Certification	93
Annex A (normative) List of codes	94
Annex B (informative) Binary examples of KNXnet/IP IP frames	103
Annex C (normative) KNXnet/IP Parameter Object.....	122

Annex D (normative) Common External Messaging Interface (cEMI)	125
Annex E (normative) Coupler Resources	158
Bibliography	170

Foreword

This document (EN 13321-2:2012) has been prepared by Technical Committee CEN/TC 247 “Building Automation, Controls and Building Management”, the secretariat of which is held by SNV.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by June 2013, and conflicting national standards shall be withdrawn at the latest by June 2013.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

This document supersedes EN 13321-2:2006.

Whereas ENV 13321-2:2000 described the transmission of EIB packets over Ethernet including the frame encoding, this document describes the transmission of HBES packets using the Internet Protocol. Details of the HBES packet frames are covered in part 1 of EN 13321, removing the need to explicitly describe the HBES frames in this document.

This document is Part 2 of the EN 13321 series of European Standards under the general title *Open data communication in building automation, controls and building management — Home and building electronic systems*, which consists of the following parts:

- *Part 1: Product and system requirements;*
- *Part 2: KNXnet/IP communication.*

According to the CEN/CENELEC Internal Regulations, the national standards organisations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Introduction

This European Standard is intended for the design of new buildings and the retrofit of existing buildings in terms of acceptable indoor environment, practical energy conservation and efficiency.

This standard defines the integration of KNX protocol implementations within the Internet Protocol (IP) named KNXnet/IP. It defines a standard protocol, which is implemented within KNX devices, Engineering Tool Software (ETS) and other implementations to support KNX data exchange over IP networks. In fact, KNXnet/IP provides a general framework, which accommodates several specialised “Service Protocols” in a modular and extendible fashion.

The KNXnet/IP standard consists of the following clauses:

- Clause 1, Overview
- Clause 2, Core Specification
- Clause 3, Device Management
- Clause 4, Tunnelling
- Clause 5, Routing
- Clause 6, Remote Diagnosis and Configuration

Additional clauses may be added to the KNXnet/IP standard in the future at which time Clause 1 “Overview” as well as Annex A will need to be updated.

KNXnet/IP supports different software implementations on top of the protocol. More specifically, these software implementations can be Building Management, Facility Management, Energy Management, or simply Data Base and SCADA (Supervision, Control and Data Acquisition) packages.

Most of these packages need to be configured for the specific user application. In order to simplify this process and cut costs for engineering, KNXnet/IP provides simple engineering interfaces, namely a description “language” for the underlying KNX system. This may be done off-line, e.g. generated as an ETS export file, or on-line by a mechanism that self-describes the underlying KNX system (reading data from the system itself).

In conjunction with the EIB/KNX-to-BACnet mapping described in EN ISO 16484-5, EIB/KNX installations can very easily be integrated into BACnet system environments.

KNXnet/IP supports:

- on-the-fly change-over between Operational modes (configuration, operation);
- event driven mechanisms;
- connections with a delay time greater than $t_{\text{EIB_transfer_timeout}}$ (e.g. network connection via satellite).

Clause 1, Overview

Clause 1 “Overview” provides a general overview of KNXnet/IP and covers security considerations.

Clause 2, Core specification

Clause 2 “Core Specification” defines a standard protocol which is implemented within KNXnet/IP devices and Engineering Tool Software to support KNX data exchange over IP networks.

This specific implementation of the protocol over the Internet Protocol (IP) is called KNXnet/IP.

This standard addresses:

- definition of data packets sent over the IP host protocol network for KNXnet/IP communication;
- discovery and self-description of KNXnet/IP servers;
- configuration and establishment of a communication channel between a KNXnet/IP client and a KNXnet/IP server.

Clause 3, Device Management

Clause 3 “Device Management” defines services for remote configuration and remote management of KNXnet/IP servers.

Clause 4, Tunnelling

Clause 4 “Tunnelling” defines services for point-to-point exchange of KNX telegrams over an IP network between a KNXnet/IP device acting as a server and a KNXnet/IP Client. This point-to-point exchange may be established by a super ordinate system for building automation or management functions or by an Engineering Tool Software. It supports all ETS functions for download, test, and analysis of KNX devices on KNX networks connected via KNXnet/IP servers. This includes changes of single KNX device object properties.

Tunnelling assumes that a data transmission round-trip between a KNXnet/IP Tunnelling client and KNXnet/IP servers takes less than $t_{\text{KNX_transfer_timeouts}}$.

Clause 5, Routing

Clause 5 “Routing” defines services for a point-to-multipoint exchange of KNX telegrams over an IP network between KNXnet/IP routers and/or KNX/IP devices.

Clause 6, Remote Diagnosis and Configuration

Clause 6 “Remote Diagnosis and Configuration” defines services for a point-to-point exchange of KNX telegrams over an IP network between KNXnet/IP routers and/or KNX/IP devices. The services provide means for diagnosing communication settings and for changing these remotely.

1 Scope

This European Standard 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.

Widespread deployment of data networks using the Internet Protocol (IP) presents an opportunity to expand building control communication beyond the local KNX control bus, providing:

- remote configuration;
- remote operation (including control and annunciation);
- fast interface from LAN to KNX and vice versa;
- WAN connection between KNX systems (where an installed KNX system is at least one line).

A KNXnet/IP system contains at least these elements:

- one EIB line with up to 64 (255) EIB devices;
OR
one KNX segment (KNX-TP1, KNX-TP0, KNX-RF, KNX-PL110, KNX-PL132);
- a KNX-to-IP network connection device (called KNXnet/IP server);

and typically additional

- software for remote functions residing on e.g. a workstation (may be data base application, BACnet Building Management System, browser, etc.).

Figure 1 shows a typical scenario where a KNXnet/IP client (e.g. running ETS) accesses multiple KNX installed systems or KNX subnetworks via an IP network. The KNXnet/IP client may access one or more KNXnet/IP servers at a time. For subnetwork, routing server-to-server communication is possible.

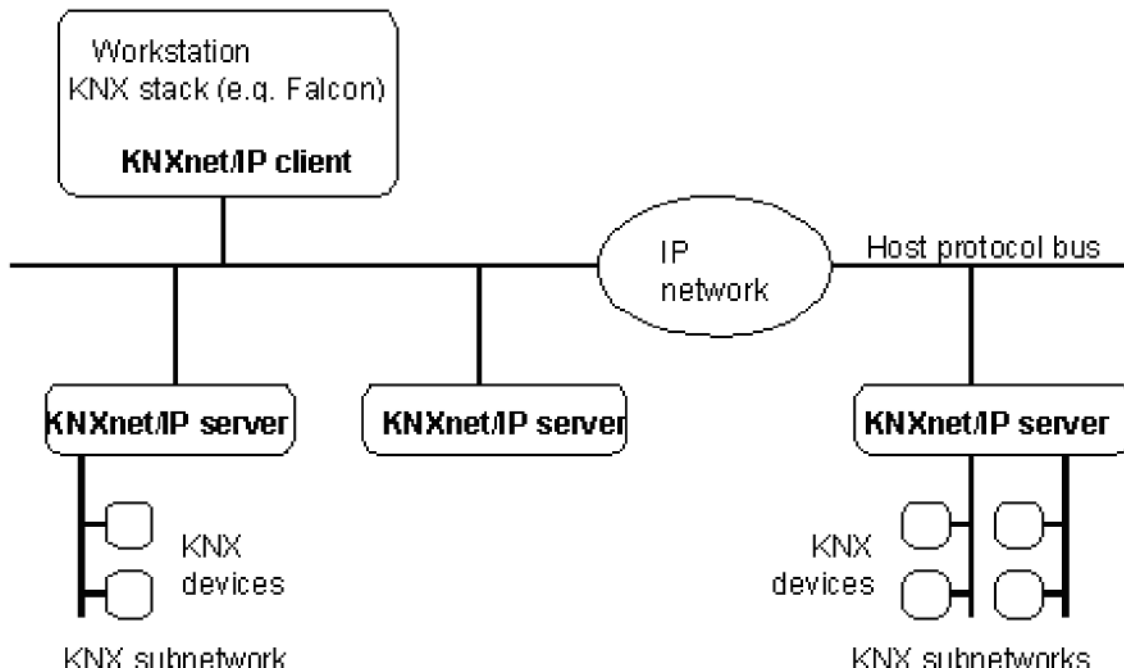


Figure 1 — Device types and configuration examples

2 Normative references

Not applicable.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1

subnet

portion of a network that shares a common address component known as the "subnet address"

Note 1 to entry: Different network protocols specify the subnet address in different ways.

3.2

Engineering Tool Software

ETS

software used to configure KNX devices

3.3

Host Protocol Address Information

HPAI

structure holding the IP host protocol address information used to address a KNXnet/IP endpoint on another KNXnet/IP device

3.4

communication channel

logical connection between a KNXnet/IP client and a KNXnet/IP server (or, in case of routing, between two or more KNXnet/IP servers)

Note 1 to entry: A communication channel consists of one or more connections on the definition of the host protocol used for KNXnet/IP.

3.5

KNX node

device implementing a KNX protocol stack and fulfilling the requirements for certification according to the KNX standard

3.6

KNXnet/IP server

KNX device with physical access to a KNX network implementing the KNXnet/IP server protocol to communicate with KNXnet/IP clients or other KNXnet/IP servers (in case of routing) on an IP network channel

Note 1 to entry: A KNXnet/IP server is by design always also a KNX node.

3.7

KNXnet/IP client

application using the KNXnet/IP client protocol to get access to a KNX subnetwork over an IP network channel

3.8

KNXnet/IP device

implementation of KNXnet/IP services on a KNX node (KNXnet/IP server) or any other hardware (KNXnet/IP client)

3.9

KNXnet/IP router

special type of KNXnet/IP device that routes KNX protocol packets between KNX sub-networks

3.10

Time To Live

TTL

maximum number of IP routers a multicast UDP/IP datagram may be routed through

Note 1 to entry: Each IP router the datagram passes decrements the TTL by one; the local host adapter also does this. When the TTL has reached zero, the router discards the datagram. When sending a datagram from the local host adapter, a TTL of zero means that the datagram never leaves the host. A TTL of one means that the datagram never leaves the local network (it is not routed).

3.11

KNXnet/IP Tunnelling

services for point-to-point exchange of KNX telegrams over an IP network between a KNXnet/IP device acting as a server and a KNXnet/IP client

3.12

Internet Control Message Protocol

ICMP

extension to the Internet Protocol (IP) for error, control, and informational messages

Note 1 to entry: ICMP is defined by RFC ¹⁾ 92 and supports packet containing error, control, and informational messages. The PING command, for example, uses ICMP to test an Internet connection.

3.13

Internet Group Management Protocol

IGMP

extension to the Internet Protocol (IP) for management of IP multicasting in the Internet

Note 1 to entry: IGMP is defined in RFC 1112 as the standard for IP multicasting in the Internet. It is used to establish host memberships in particular multicast groups on a single network. By using Host Membership Reports, the

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

mechanisms of the protocol allow a host to inform its local router that it wants to receive messages addressed to a specific multicast group. All hosts conforming to level 2 of the IP multicasting specification require IGMP.

3.14 **IP channel**

logical connection between two IP host/port endpoints

Note 1 to entry: IP channels are either a guaranteed, reliable TCP (transmission control protocol) or an unreliable point-to-point or multicast (in case of routing) UDP (user datagram protocol) connection.

3.15 **communication channel**

as defined by the KNXnet/IP Core specification, this is represented by one or two IP channels

3.16 **common External Message Interface** **cEMI**

generic structure for medium independent KNX messages

Note 1 to entry: cEMI (common EMI) frames are used to encapsulate KNX messages within Internet Protocol (IP) packets.

4 Symbols and abbreviations

For the purpose of this document, the symbols, abbreviations and acronyms used are listed below.

Tables listing implementation requirements use the following abbreviations.

Symbol	Description
M	Mandatory
C ⁿ	Conditions are specified under note "n"
O	Optional
X	Not allowed
n/a	Not applicable
R	Required
MC	Message code
AddIL	length of additional information

4.1 **DHCP** **Dynamic Host Configuration Protocol**

communication protocol for automatic assignment of IP address settings

4.2 **DNS** **Domain Name Service**

assigns Internet names to IP addresses

4.3 **EIB** **European Installation Bus**

Standard for Building Controls (EN 50090)

4.4 **IP** **Internet Protocol**

4.5 **KNX** Standard for Building Controls (EN 50090)

4.6 **TCP/IP** **Transmission Control Protocol over Internet Protocol** connection-oriented communication over the Internet

4.7 **UDP/IP** **User Datagram Protocol over Internet Protocol** connection-less communication over the Internet

5 Requirements

5.1 Clause 1: Overview

5.1.1 KNXnet/IP Document Clauses

5.1.1.1 General

This European Standard defines the integration of KNX protocol implementations within the Internet Protocol (IP) named KNXnet/IP or EIBnet/IP for continuity with ENV 13321-2:2000 (EIBnet) as defined by CEN/TC 247. EIBnet was introduced as an expansion of EIB into the information technology realm and was incorporated as a building controls technology in CEN/TC 247. EIBnet/IP is the logical successor to EIBnet. As EIB has become a part of KNX, this standard is called KNXnet/IP.

This European Standard defines a standard protocol, which is implemented within KNX devices, Engineering Tool Software and other implementations to support KNX data exchange over IP networks. In fact, KNXnet/IP provides a general framework, which accommodates several specialised “Service Protocols” in a modular and extendible fashion.

The KNXnet/IP standard consists of the following clauses:

- Clause 1, Overview;
- Clause 2, Core Specification;
- Clause 3, Device Management;
- Clause 4, Tunnelling;
- Clause 5, Routing.

Additional clauses may be added to the KNXnet/IP standard in the future at which time this Clause 1 “Overview” shall be updated.

KNXnet/IP supports different software implementations on top of the protocol. More specifically, these software implementations can be Building Management, Facility Management, Energy Management, or simply Data Base and SCADA (Supervision, Control and Data Acquisition) packages.

Most of these packages need to be configured for the specific user application. In order to simplify this process and cut costs for engineering, KNXnet/IP provides simple engineering interfaces, namely a

description “language” for the underlying KNX system. This may be done off-line, e.g. generated as an ETS export file, or on-line by a mechanism that self-describes the underlying KNX system (reading data from the system itself).

KNXnet/IP supports:

- on-the-fly change-over between operational modes (configuration, operation);
- event driven mechanisms;
- connections with a delay time greater than $t_{\text{ElB_transfer_timeout}}$ (e.g. network connection via satellite).

5.1.1.2 Clause 1, Overview

Clause 1 "Overview" is this document.

5.1.1.3 Clause 2, Core specification

Clause 2 “Core Specification” defines a standard protocol, which is implemented within KNXnet/IP devices and the Engineering Tool Software (ETS) to support KNX data exchange over IP networks.

This specific implementation of the protocol over the Internet Protocol (IP) is called KNXnet/IP.

This standard addresses:

- definition of data packets sent over the IP host protocol network for KNXnet/IP communication;
- discovery and self-description of KNXnet/IP servers;
- configuration and establishment of a communication channel between a KNXnet/IP client and a KNXnet/IP server;

5.1.1.4 Clause 3, Device Management

Clause 3 “Device Management” defines services for remote configuration and remote management of KNXnet/IP servers.

5.1.1.5 Clause 4, Tunnelling

Clause 4 “Tunnelling” defines services for point-to-point exchange of KNX telegrams over an IP network between a KNXnet/IP device acting as a server and a KNXnet/IP Client. This point-to-point exchange may be established by a super ordinate system for building automation or management functions or by an Engineering Tool Software. It supports all ETS functions for download, test, and analysis of KNX devices on KNX networks connected via KNXnet/IP servers. This includes changes of single KNX device object properties.

Tunnelling assumes that a data transmission round-trip between ETS or a KNXnet/IP Tunnelling client and KNXnet/IP servers takes less than $t_{\text{KNX_transfer_timeouts}}$.

5.1.1.6 Clause 5, Routing

Clause 5 “Routing” defines services, which route KNX telegrams between KNXnet/IP servers through the IP network.

5.1.2 Mandatory and optional implementation of IP protocols

5.1.2.1 General

KNXnet/IP uses existing IP protocols where possible unless their use implies an undue burden with regard to memory and implementation requirements for the intended service.

The following table shows mandatory (M) and optional (O) implementation of IP protocols by KNXnet/IP service types. Although this table refers to the KNXnet/IP server, it also indicates which IP protocols shall be implemented by the KNXnet/IP client. Any non-applicable IP protocol is marked as "na".

Table 1 — KNXnet/IP service types and IP protocols

IP protocol	Service Type			
	Core	Device Management	Tunnelling	Routing
ARP	M	M	M	M
RARP	O	O	O	O
Support of fixed IP address	M	M	M	M
BootP (Client) ^a	M	M	M	M
DHCP (Client) ^a	M	M	M	M
UDP	M	M	M	M
TCP	O	O	O	na
ICMP	M	M	M	M
IGMP	M	M	na	M
^a BootP/DHCP: It is essential that either one be implemented by a KNXnet/IP device.				

Other Internet protocols like NTP (network time protocol), FTP (file transfer protocol), HTTP (hypertext transfer protocol), SMTP (simple message transfer protocol), DNS (domain name system), and SNMP (simple network management protocol) may be used but are not within the scope of the KNXnet/IP protocol.

5.1.2.2 Minimum KNXnet/IP device requirements

KNXnet/IP service types as defined in this standard require the implementation of a minimal set of IP protocols for interworking.

KNXnet/IP servers shall implement these IP protocols: ARP, BootP, UDP, ICMP and IGMP. Other IP protocols may be required for specific services.

5.1.2.3 Network environment

Because KNXnet/IP servers use IP, this standard does not require any specific medium carrying the IP datagrams.

It is RECOMMENDED to use a medium, which carries at least twice the bit rate of all KNXnet/IP routers connected to this medium. For a point-to-point, e.g. PPP, connection this would be at least 19 200 bit/s; for a network with up to 400 KNXnet/IP servers, this would be least 8 Mbit/s.

10BaseT is RECOMMENDED as a minimum for KNXnet/IP servers using Ethernet as physical medium.

5.1.2.4 Addressing

KNXnet/IP servers are typically connected to one KNX subnetwork and to an IP network. Hence, KNXnet/IP servers have one distinct address for each network they are connected to: one KNX/EIB Individual Address and one IP address.

Additionally, KNXnet/IP servers are assigned to a KNXnet/IP project-installation using the same IP multicast address for all KNXnet/IP servers in a KNXnet/IP project-installation.

5.1.2.5 KNXnet/IP device classes

A KNXnet/IP device can be a software implementation running on a PC. Hence, ETS or any other software implementing KNXnet/IP services is viewed as a KNXnet/IP device.

Definition of KNXnet/IP device classes ensures interoperability between KNXnet/IP devices as well as a minimum set of supported KNXnet/IP services for a given KNXnet/IP device class.

Table 2 lists mandatory and optional service types for device classes.

Device Class A encompasses configuration and system maintenance tools. Except for Object Server services, all other KNXnet/IP services shall be implemented. ETS is a member of this device class.

Device Class B defines the minimum set of KNXnet/IP services for KNXnet/IP routers.

Device Class C defines the minimum set of KNXnet/IP services for any other KNXnet/IP device. Building, energy and facilities management systems are members of this KNXnet/IP device class.

Table 2 — KNXnet/IP device classes

Device Class	Service Type			
	Core	Device Management	Tunnelling	Routing
A (Configuration and System Maintenance Tools)	M	M	M	M
B (KNXnet/IP Router)	M	M	M	M
C (any other KNXnet/IP device)	M	M	O	O

5.1.3 Security considerations

5.1.3.1 Introduction

For KNX, security was and is a minor concern, as any breach of security requires local access to the network. In the case of KNX TP (EIB), and KNX PL networks, this requires even physical access to the network wires, which in nearly all cases is impossible as the wires are inside a building or buried underground.

Hence, security aspects are less of a concern for KNX field level media.

The intention of KNXnet/IP is to provide a means of using existing data networking technology, i.e. the Internet Protocol, for fast communication between different KNX subnetworks in the same or different locations.

Unless the data network is completely isolated and only used for KNXnet/IP, a number of potential threats need to be considered.

5.1.3.2 Categories of threats

5.1.3.2.1 General

This section briefly enumerates various types of security threats and mentions typical countermeasures.

5.1.3.2.2 Passive attacks

Passive attacks involve a violation of privacy or leak of information to an unauthorised party that “wiretaps” or listens in on the network. Passive attacks are difficult to detect, since they do not modify the network traffic in any way. The primary defence against passive attacks is isolation of network traffic (one cannot eavesdrop or analyse traffic that one cannot see):

- Eavesdropping – interception and examination of packet contents by unauthorised third parties. In addition to network isolation, this threat can be countered by encrypting packets.
- Traffic analysis – interception and analysis of traffic characteristics without examining the contents of individual packets. This threat exists even if packets are encrypted, and it is very difficult to counter except by isolation of network traffic.

5.1.3.2.3 Active attacks

Active attacks involve some type of modification to network traffic: adding, deleting, or delaying packets, or changing the contents of packets in some way. Types of active attack are listed below:

- Masquerade – unauthorised entities forging traffic to make it appear to originate from a legitimate source. This threat can be countered by authentication mechanisms.
- Access control violation – a user accesses a resource that they do not have proper authorisation to use. This threat can be countered by message integrity, authentication, and authorisation mechanisms (the access control policies shall also be secured).
- Modification – intercepting and changing the contents of packets on the network. This threat can be countered by message integrity mechanisms.
- Deletion – preventing packets from arriving at their intended destination (for example, by maliciously configuring a router to drop packets or by deliberately interfering with packet transmissions on the wire). This threat can be very difficult to counter, especially if the attacker can jam packet transmissions on the physical network.

- Delay – delaying arrival of packets at their intended destination. This attack can be achieved by compromising or overloading a router or bridge or by performing a time-limited attack on the packet transmission (thereby causing retransmission delays). This threat is also very difficult to counter.
- Replay – intercepting packets and later resending them on the network. This threat can be countered by authentication, message integrity, and timestamps or message counters.
- Denial of service – this type of attack typically involves injecting a flood of useless traffic on the network to consume resources such as bandwidth or processing time so that the resources are unavailable to service legitimate traffic. This threat is difficult to counter, especially if the attack traffic cannot be easily distinguished from legitimate traffic.

5.1.3.3 Countermeasures

The following is a list of countermeasures for KNXnet/IP:

- a) Typically, building control data is only exchanged in a closed or tightly surveillance network. Use KNXnet/IP in an Intranet or over Virtual Private Network only.
- b) Filtering KNXnet/IP datagrams from the network requires network analysis tools and expertise. The content of a KNXnet/IP message is not self-descriptive but requires semantic knowledge residing in ETS. Access to ETS is limited to authorised personnel only.
- c) Use authentication when opening point-to-point connections e.g. for tunnelling or remote logging.
- d) Do not publish default KNXnet/IP IP multicast address.
- e) Use sequence counter on all point-to-point connections e.g. for tunnelling or remote logging.
- f) Use UDP/IP or TCP/IP port numbers, which are filtered by a firewall.

Table 3 — Threats and countermeasures

Threat...	Is countered by measure(s)
Eavesdropping	a), b), d)
Traffic analysis	a), d)
Masquerade	a), c), d)
Access control violation	a), c), d)
Modification	a), e)
Deletion	a)
Delay	a)
Replay	c), e)
Denial of service	a)

5.1.3.4 Conclusion

Most of the listed threats are countered by staying within a network, which tightly supervises and restricts network access from outside the network.

Using authentication when establishing point-to-point connections is an additional means to this end.

It is quite unlikely that legitimate users of a network would have the means to intercept, decipher, and then tamper with the KNXnet/IP without excessive study of the KNX standards.

Thus the remaining security threat is considered to be very low and does not justify mandating encryption, which would require considerable computing resources.

5.2 Clause 2: Core

5.2.1 Scope

This Clause "Core" of the KNXnet/IP standard provides a general host protocol-independent framework, which accommodates several specialised "Service Protocols" in a modular and extendible fashion.

This standard addresses:

- definition of data packets sent over the non-KNX "host protocol" network for KNXnet/IP communication;
- discovery and self-description of KNXnet/IP servers; and
- configuration, establishment and maintenance of a communication channel between a KNXnet/IP client and a KNXnet/IP server.

5.2.2 KNXnet/IP frames

5.2.2.1 General definitions

5.2.2.1.1 Data format

The KNXnet/IP frames described within this standard are coded binary.

The data structure specifications are always noted in binary format.

5.2.2.1.2 Byte order

For binary structures, if not explicitly stated otherwise, the byte order shall be big endian mode (Motorola, non-swapped). For plain text formats, the byte order and formatting shall be according to the related protocol specifications.

5.2.2.1.3 Structures

All KNXnet/IP structures follow a common rule: the first octet is always the length of the complete structure (as some structures may have fields of variable length e.g. strings) and the second octet is always an identifier that specifies the type of the structure. From the third octet on, the structure data follows. If the amount of data exceeds 252 octets, the length octet shall be FFh and the next two octets will contain the length as a 16 Bit value. Then the structure data starts at the fifth octet.

5.2.2.2 Frame format

The communication between KNXnet/IP devices is based on KNXnet/IP frames. A KNXnet/IP frame is a data packet sent over the non-KNX network protocol that consists of a header, comparable to the IP header of an internet protocol data packet and an optional body of variable length.

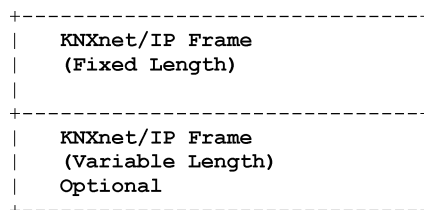


Figure 2 — KNXnet/IP frame binary format

The type of KNXnet/IP frame is described by a KNXnet/IP service type identifier in the header. KNXnet/IP services include, but are not limited to, information regarding discovery and description, communication channel (connection) management and KNX data transfer.

5.2.2.3 Header

5.2.2.3.1 Description

Every KNXnet/IP frame, without any exception, consists at least of the common KNXnet/IP header that contains information about the protocol version, the header and total packet length and the KNXnet/IP service type identifier. The KNXnet/IP header may be followed by a KNXnet/IP body, depending on the KNXnet/IP service.

Timestamp information and frame counters are not included in the common KNXnet/IP frame header as this information is closely linked with certain KNXnet/IP service types and will therefore be included in the body of these services as additional information for certain communication channel types.

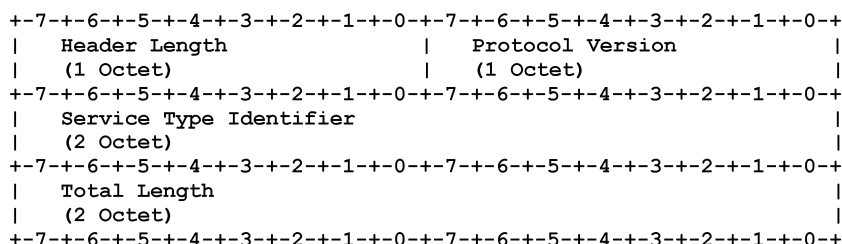


Figure 3 — KNXnet/IP header binary format

5.2.2.3.2 Header length

Although the length of the header is always fixed, it is possible that the size of the header changes with a new version of the protocol. The header length can be used as an index into the KNXnet/IP frame data to find the beginning of the KNXnet/IP body.

5.2.2.3.3 Protocol version

The protocol version information states the revision of the KNXnet/IP protocol that the following KNXnet/IP frame is subject to. It will be stored in binary coded decimal format. The only valid protocol version at this time is 1.0 (represented as hexadecimal 10h).

5.2.2.3.4 KNXnet/IP service

The KNXnet/IP service type identifier defines the kind of action to be performed and the type of the data payload contained in the KNXnet/IP body if applicable. The high octet of the KNXnet/IP service type identifier denotes the service type family and the low octet the actual service type in that family. For a detailed description of the services, see below.

5.2.2.3.5 Total length

The total length is expressing the total KNXnet/IP frame length in octets. The length includes the complete KNXnet/IP frame, starting with the header length of the KNXnet/IP header and including the whole KNXnet/IP body.

5.2.3 Host protocol independence

5.2.3.1 Host protocol

The KNXnet/IP core specification defines the integration of KNX protocol implementations on top of the Internet Protocol (IP). It describes the general and host protocol independent as well as the host protocol specific parts of the KNXnet/IP communication.

5.2.3.2 Endpoints

KNXnet/IP defines the Host Protocol Address Information (HPAI) structure, which is the combination of IP address and port number. The HPAI is the data required to send a KNXnet/IP frame to another device. The KNXnet/IP standard uses the term KNXnet/IP endpoint as a logical view of a HPAI to address another KNXnet/IP device for certain well-defined purposes:

Every KNXnet/IP device shall support exactly one device related, bi-directional and connectionless endpoint for discovery if the host protocol requires discovery services. It shall support at least one bi-directional and connectionless endpoint for controlling and at least one bi-directional and connection oriented endpoint for service type related data transmission.

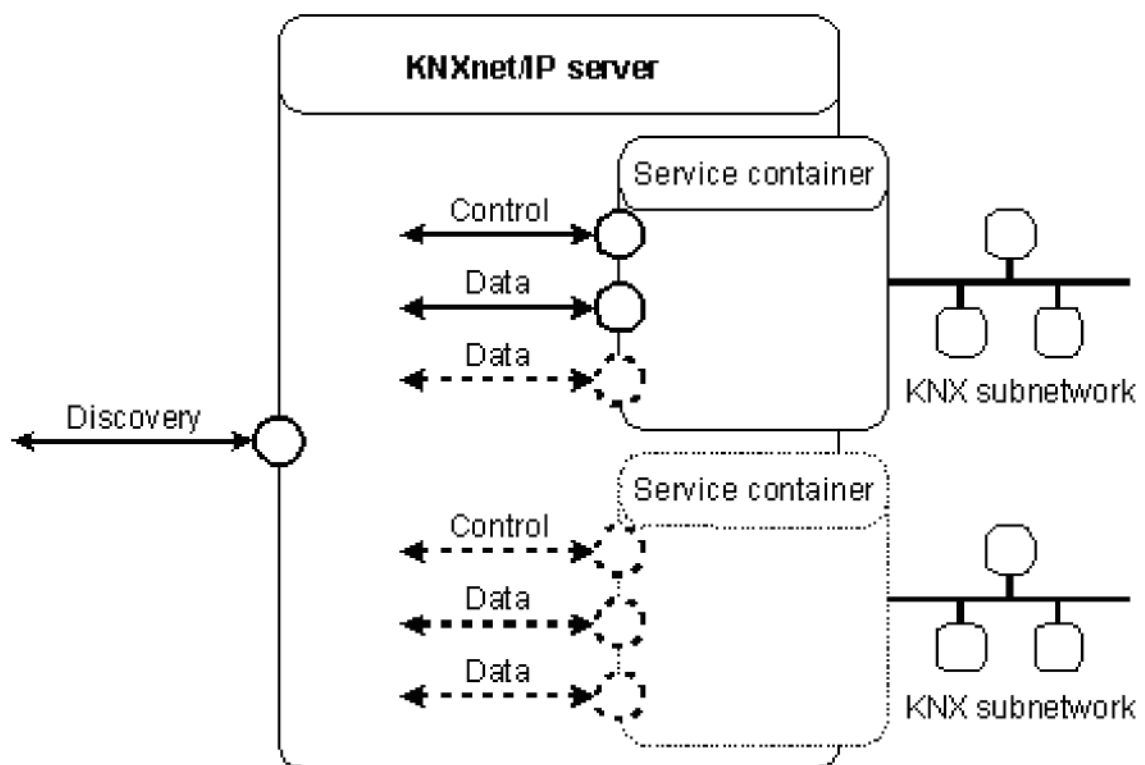


Figure 4 — KNXnet/IP server endpoints sample configuration

The control endpoint uniquely addresses one entity inside the KNXnet/IP server device that shall be capable of providing at least one KNXnet/IP service type.

This entity, called service container, may be connected to a KNX Subnetwork. If the KNXnet/IP server device supports more than one KNX Subnetwork connection, it is REQUIRED that every KNX Subnetwork shall be

represented by a different control endpoint. The KNXnet/IP client therefore considers 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 server devices.

These KNXnet/IP endpoints present a logical view to the communication of a KNXnet/IP device. The actual implementation of these endpoints with different host protocols may use transport medium dependent approaches that differ from this logical view. For example, the bi-directional KNXnet/IP endpoints could be implemented using two unidirectional channels with the host protocol. Therefore, it is possible for one KNXnet/IP endpoint to be represented by multiple HPAs.

5.2.4 Discovery and self description

5.2.4.1 Introduction

Particularly for networks supporting hot-plug, and where even the address assignment may take place at runtime (e.g. IP address assignment via BootP or DHCP), it is of significant importance to search for devices within a subnetwork without having the need to retrieve network parameters through a non-standardised way and manually input them in the client tool to establish a connection. Furthermore, to get a precise picture of the services supported by the KNXnet/IP server without implementing trial and error, a self-description mechanism is an important feature.

5.2.4.2 Discovery

Any KNXnet/IP server shall implement discovery according to this procedure. If applicable for the host protocol, it is recommended that a KNXnet/IP client implementation supports searching for KNXnet/IP servers instead of requiring manual input.

The discovery operation consists of a SEARCH_REQUEST data packet, sent via multicast using the discovery endpoint, which contains the HPAI of the KNXnet/IP client's discovery endpoint. The HPAI may contain a unicast IP address to receive the answers from the different KNXnet/IP servers directly in a point-to-point manner. Typically, it should contain the KNXnet/IP System Setup multicast address to ensure reception from KNXnet/IP servers that are on a different subnetwork.

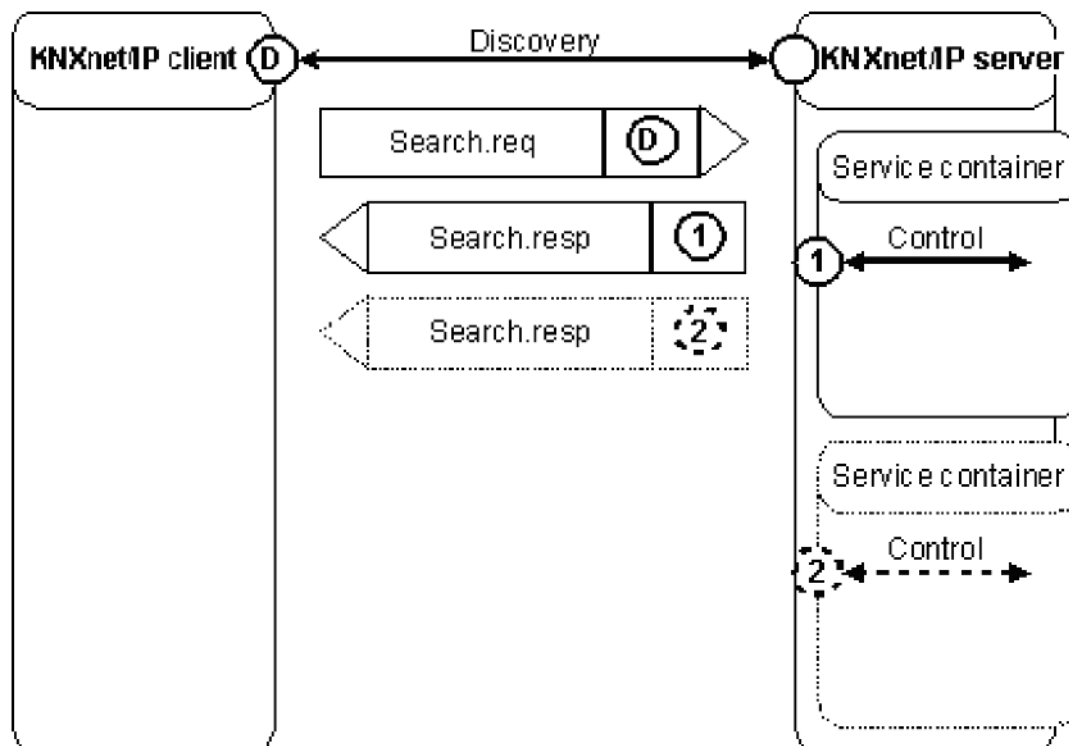


Figure 5 — Discovery procedure

After sending the request, the KNXnet/IP client shall wait for time SEARCH_TIMEOUT for SEARCH_RESPONSE frames from KNXnet/IP servers. After that period of time, any received SEARCH_RESPONSE frame shall be ignored by that client until it starts another discovery cycle. SEARCH_REQUEST frames received by clients from other clients shall be ignored.

Any KNXnet/IP server receiving a SEARCH_REQUEST service shall respond immediately with a SEARCH_RESPONSE data packet to the given HPAI using its discovery endpoint. Such a response contains only the HPAI of the KNXnet/IP server's control endpoint for all further communication.

Any KNXnet/IP server shall support discovery by processing search requests and sending correct responses. A KNXnet/IP server may support links to more than one KNX Subnetwork, it shall however send a SEARCH_RESPONSE data packet for the control endpoint of each KNX Subnetwork it supports connections to, even if it supports only one data connection at a time.

5.2.4.3 Self-description

Typically, after discovering a KNXnet/IP server, the KNXnet/IP client sends a DESCRIPTION_REQUEST through a unicast or point-to-point connection to all control endpoints of the KNXnet/IP server. It is REQUIRED that every KNXnet/IP implementation supports description requests. Furthermore, before a KNXnet/IP client communicates with a KNXnet/IP server, it should check if the server supports the services requested by the client using the self-description mechanism.

If a KNXnet/IP server receives a valid description request, it shall reply with a DESCRIPTION_RESPONSE packet providing information on the supported protocol version range, its own capabilities, state information and optionally a friendly name for this KNXnet/IP server's KNX connection. As a KNXnet/IP server may support links to more than one KNX Subnetwork, it shall support responding to discovery requests for each potential KNX Subnetwork connection announced by the discovery responses.

5.2.5 Communication Channels

5.2.5.1 Introduction

A communication channel is the data endpoint connection between a KNXnet/IP client and a KNXnet/IP server. Data endpoint connections are established for services requiring point-to-point communication e.g. tunnelling or device management. Any point-to-point connection between a KNXnet/IP client and a KNXnet/IP server shall be initiated by the client. Any KNXnet/IP server shall support at least one client connection at a time. It may support more than one client connection at a time, however it shall ensure that existing connections are not affected by accepting new connections (e.g. a KNXnet/IP server shall not accept a tunnelling connection on the same physical access to a KNX Subnetwork in different modes, link or busmonitor layer).

5.2.5.2 Establishing a link

To establish a link between a KNXnet/IP client and a KNXnet/IP server, the client shall send a CONNECT_REQUEST frame to the control endpoint of the server. This request provides information on the requested connection type (e.g. data tunnelling or remote logging), general and connection type specific options (e.g. link layer or busmonitor mode²⁾) and the data endpoint HPAI that the client wants to use for this communication channel.

Before sending a connection request of a specific type (with specific options), the KNXnet/IP client should check against the self-description information received from the KNXnet/IP server if the server supports the requested connection type and/or all the requested options.

The KNXnet/IP server shall then send a CONNECT_RESPONSE frame in any case back to the KNXnet/IP client requesting to establish the connection, providing the status of the request (ACK/NACK with extended status information if applicable). If the request could be accepted by the server, the CONNECT_RESPONSE

2) The list of supported layers and services is supposed to be extended in further versions.

frame contains additionally an identifier as well as the HPAI of the data endpoint that the server now prepared for this communication channel³⁾

After sending the connection request, the KNXnet/IP client shall wait for the host protocol dependent time `CONNECT_REQUEST_TIMEOUT` (= 10 s) for the response frame from KNXnet/IP server. After that period of time, any received response frames shall be ignored by that client until it starts another connection request.

The current protocol standard assumes that a connection shall not be shared by multiple clients. A KNXnet/IP server shall therefore not accept multiple connect requests of the same type on e.g. the same physical KNX connection, though it may of course implement numerous physical connections, exposing each logically as an independent KNXnet/IP server, if supported by the used host protocol. The client implementation can rely on that restriction and is not required to handle such a connection sharing scenario. Service family exceptions to this general rule are described in the corresponding service family KNXnet/IP chapter.

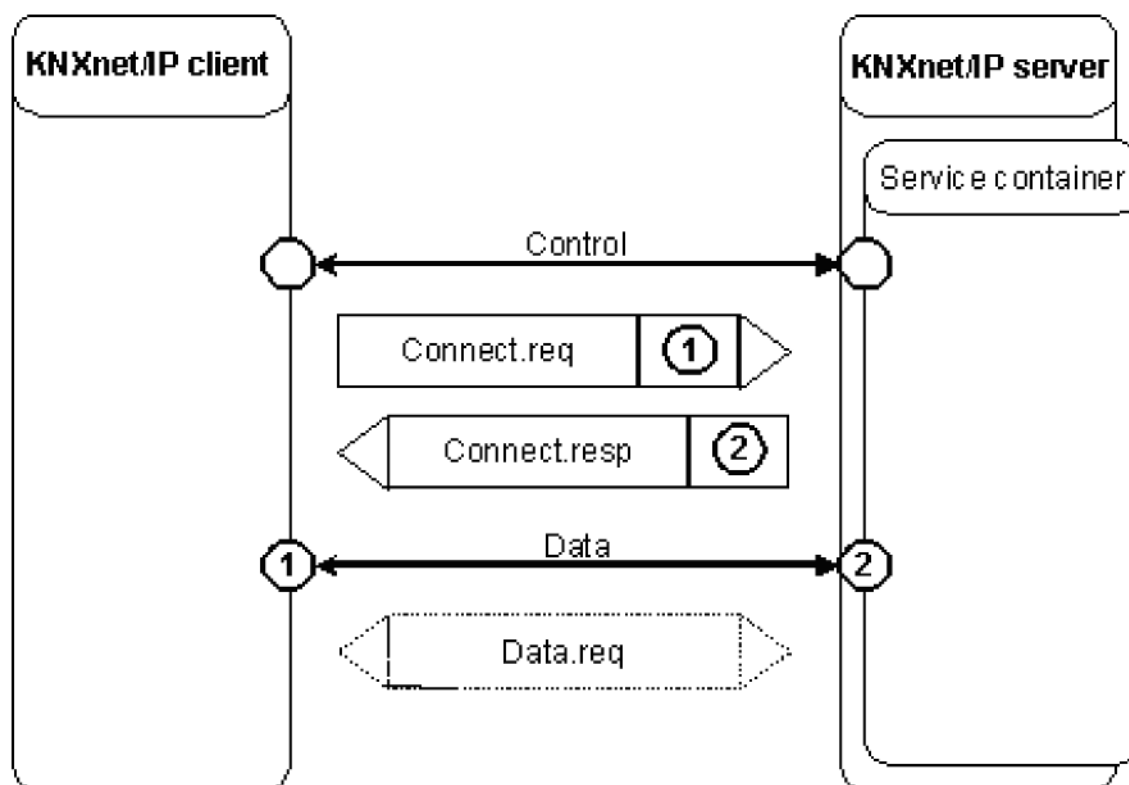


Figure 6 — Establishing a data connection

5.2.5.3 Connection Header

5.2.5.3.1 Description

The body of every KNXnet/IP frame sent over an established communication channel starts with data fields that contain additional general information about the data connection. The amount of this data and what type of information is included is determined by several options during the connection phase of a communication channel. The total of these data fields is called connection header and its appearance varies greatly depending on the already mentioned connection options. Only the order in which the different data fields are stored in the connection header is fixed.

³⁾ It should be noted that an EIBnet/IP connection may consist due to technical reasons of multiple logical connections at the host protocol layer.

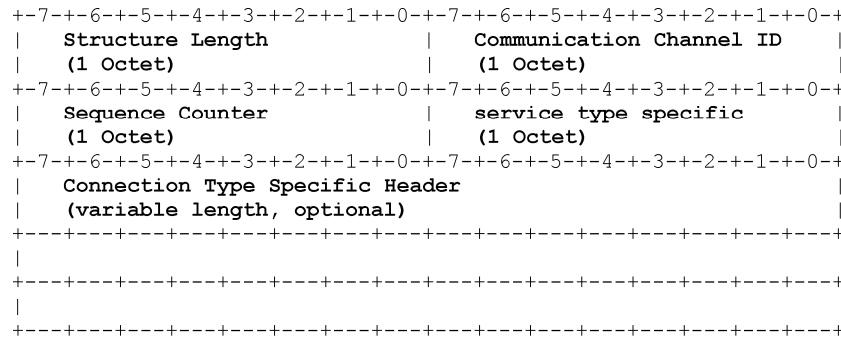


Figure 7 — Common connection header

5.2.5.3.2 Structure length

Structure Length is the total length of the connection header.

5.2.5.3.3 Communication Channel ID

The KNXnet/IP server assigns a Communication Channel ID to each established communication channel.

5.2.5.3.4 Sequence Counter

The sequence counter is maintained for each communication channel. It shall be incremented by one independently for every communication channel for each KNXnet/IP frame sent over the data connection. Both KNXnet/IP devices maintaining the communication channel have independent sequence counters.

Every time a connection is established, the counter for this connection shall be set to 0, so the first KNXnet/IP frame sent on an established communication channel has a sequence counter of 0.

5.2.5.3.5 Connection Type specific Header Items

The connection type specific header items are optional and of variable length depending on the type of the connection.

5.2.5.4 Heartbeat monitoring

Host protocols not providing mechanisms for lifetime check like UDP/IP need a procedure to identify failure of communication, may it be on the KNX or the tunnelling network. To detect such situations heartbeat monitoring is defined and shall be implemented by both KNXnet/IP clients and servers.

The client shall send a CONNECTIONSTATE_REQUEST frame regularly, i.e. every 60 s, to the server's control endpoint which shall respond immediately with a CONNECTIONSTATE_RESPONSE frame (this counts as heartbeat response).

If the KNXnet/IP client does not receive the heartbeat response within the CONNECTIONSTATE_REQUEST_TIMEOUT (= 10 s) or the status of a received heartbeat response signalled any kind of error condition, the client shall repeat the CONNECTIONSTATE_REQUEST three times and then terminate the connection by sending a DISCONNECT_REQUEST to the server's control endpoint.

If the KNXnet/IP server does not receive a heartbeat request within 120 s of the last correctly received message frame, the server shall terminate the connection by sending a DISCONNECT_REQUEST to the client's control endpoint. The server shall not retrigger the timeout after messages received with unexpected sequence number.

5.2.5.5 Disconnecting

Typically, the client terminates the connection. During normal operation, the client shall send a DISCONNECT_REQUEST to the server's control endpoint to request termination of the data channel connection.

The client should try to disconnect gracefully if possible, even under error conditions. The server may disconnect from the client by sending a DISCONNECT_REQUEST in case of internal problems or if it receives invalid data packets; however it is recommended to let the client terminate the connection.

The KNXnet/IP device receiving the DISCONNECT_REQUEST from the communication partner shall acknowledge the operation with a DISCONNECT_RESPONSE frame. This data packet signals the final termination of a previously established communication channel.

5.2.6 General implementation guidelines

5.2.6.1 Introduction

This section defines programming guidelines which must be taken into account when implementing KNXnet/IP servers or clients, respectively. Compliance with these guidelines is a requirement for the certification of the protocol implementation.

5.2.6.2 KNXnet/IP servers

- If a server receives a data packet with an unsupported version field, it shall reply with a NACK message indicating in the status field E_VERSION_NOT_SUPPORTED.
- If an invalid data packet is received, the implementation shall ignore the data packet without taking any further action.
- If a connection is established, all data packets shall be sent with the same protocol version.
- If a connection is established and the protocol version changes within the received data packets, the server shall shut down the connection.

5.2.6.3 KNXnet/IP clients

- If a client receives a data packet with an unsupported version field, it shall reply with a NACK message indicating in the status field E_VERSION_NOT_SUPPORTED. If a connection to that server sending with an unsupported protocol version is established, the client shall disconnect. The client may try to reconnect then and re-establish the connection.
- If an invalid data packet is received, the implementation shall ignore the data packet without taking any further action.
- If a connection is established, all data packets shall be sent with the same protocol version.
- If a connection is established and the protocol version changes within the received data packets, the client shall disconnect from the server. The client may try to reconnect then and re-establish the connection.

5.2.6.4 KNXnet/IP router settings

5.2.6.4.1 KNXnet/IP router factory default settings

An important feature for KNXnet/IP routers is that they shall provide proper KNXnet/IP Routing without any user intervention. This plug and play Routing behaviour requires a standardised factory default configuration.

- Routers shall be shipped with a default Individual Address of FF00h.
- The Routing multicast address equals the System Setup multicast address.
- KNX broadcast telegrams shall be routed from one KNX Subnetwork to another even if KNXnet/IP Routing devices are still being used in their factory default configuration.
- KNXnet/IP Routing shall already work even if a valid unicast IP address has not been obtained by or assigned to the KNXnet/IP Routing device.
- Routing shall already work and Tunnelling shall be available for (further) configuration of the KNXnet/IP router if an Individual Address has been assigned to a KNXnet/IP router via KNX Subnetwork.

5.2.6.4.2 KNXnet/IP router IP address assignment

KNXnet/IP routers shall support plug and play KNXnet/P routing out of the box even if a valid unicast IP address was not acquired from a DHCP server, by manual input, or via ETS configuration. This may require that the IP stack can send and receive multicast IP messages although a valid unicast IP address has not been acquired.

If an IP stack does not support multicasting without an assigned unicast IP address then the KNXnet/IP router shall acquire a unicast IP address via AutoIP or by self-assigning the default KNXnet/IP source unicast IP address 0.0.0.0.

If a valid unicast IP address was not acquired the KNXnet/IP router shall use the default KNXnet/IP source unicast IP address 0.0.0.0 for Routing but shall not support KNXnet/IP Tunnelling.

5.2.6.5 Initial setup procedures for KNXnet/IP Servers

5.2.6.5.1 General

KNXnet/IP devices shall be configurable in the same way as traditional KNX devices. In an unconfigured state KNXnet/IP routers shall operate with default values enabling KNX telegrams to pass from one KNX Subnetwork to another, enabling KNXnet/IP routers to transparently replace KNX routers (Line and Backbone Couplers).

This requirement is set under the condition that KNX Individual Addresses are unique across the IP network. If KNXnet/IP devices of two independent installations are connected to the same IP network or a KNX project consists of multiple installations, the use of one factory default routing configuration cannot guarantee the delivery of KNX telegrams to the intended destination as the same KNX (Individual or Group) Address could be present in different installations.

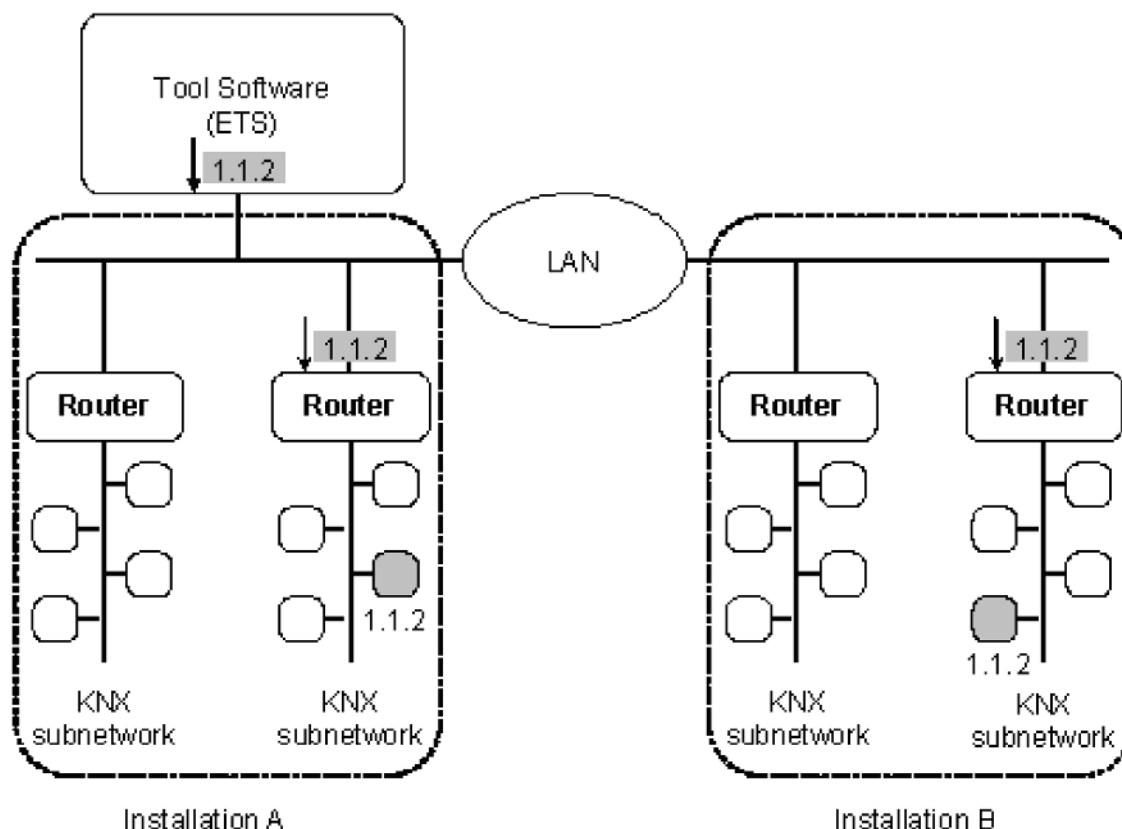


Figure 8 — KNX project with multiple installations

Under the precondition that only one installation is connected to the IP network the following rules apply:

- 1) Upon power-up, the KNXnet/IP device steps through the procedure described in KNXnet/IP Core, 5.2.8.5, IP Address Assignment, to retrieve an individual IP address.
- 2) If connected to a KNX Serial Interface device (RS232) or KNX USB Interface, the tool software can only access KNX devices on the local KNX subnetwork (including the local KNXnet/IP router) as long as not all the KNXnet/IP router devices belonging to one project are successfully configured.
- 3) The tool software **SHOULD** always attempt to configure KNXnet/IP router devices first to gain access to KNX devices behind possibly still unconfigured KNXnet/IP routers.
- 4) If directly connected to the IP network, the tool software can access all KNXnet/IP devices and through configured routers the corresponding KNX Subnetworks.
- 5) The tool software **SHOULD** use the IP network for the setup of projects containing KNXnet/IP devices.
- 6) It is possible to completely configure KNXnet/IP devices in one single configuration procedure (KNX Individual Address assignment and parameter download).
- 7) Assignment of KNX Individual Address: If the tool software is connected through a KNX Serial Interface device (RS232) or KNX USB Interface to the KNX network, the traditional KNX management procedure applies for KNXnet/IP devices as well as other KNX nodes. Devices behind unconfigured routers are not reachable. If the tool software is connected to the IP network, the configuration procedure for KNXnet/IP devices (described below) shall apply. Before setting up KNX field media devices the tool software shall firstly configure all KNXnet/IP router devices.

- 8) Parameter download: if the tool software is connected through a serial interface to the KNX network, the traditional KNX management procedure applies for KNXnet/IP devices as well as other KNX nodes. If the tool software is connected to the IP network, it establishes management connections to every KNXnet/IP device to download the device parameters (see configuration procedure). After having configured all KNXnet/IP router devices, the tool software can access all other KNX nodes of the project for further set-up and download through a KNXnet/IP tunnelling connection.
- 9) If the SEARCH_RESPONSE answers reveal that KNXnet/IP devices use different IP address spaces, the tool software shall present an error message.
- 10) If the SEARCH_RESPONSE answers reveal that the IP address of a KNXnet/IP device is different from the network settings of the tool software and the IP address of the KNXnet/IP device is an AutoIP address, the tool software shall present an error message and user control button that enables to reset the KNXnet/IP device via KNXnet/IP Routing with the KNX connectionless restart service.

5.2.6.5.2 Configuration Procedure for configuration via KNXnet/IP Routing

The tool software is connected to a KNX Subnetwork (1.1) via a KNX Serial Interface device (RS232) or KNX USB interface. A second KNX Subnetwork (1.2) is connected with the first via two KNXnet/IP routers and a LAN. All devices including the routers are initially unconfigured.

The tool software uses KNX management procedures and (indirectly) KNXnet/IP Routing to configure the two KNXnet/IP routers.

This is the configuration procedure for assignment of the KNX Individual Address and configuration of parameters of a KNXnet/IP device:

- 1) The tool software requests the user to activate the programming mode of the KNXnet/IP device.

NOTE This can be done by pressing the Programming Button on the KNXnet/IP device.

- 2) The tool software sends a KNX read (broadcast). The KNXnet/IP router on this KNX Subnetwork sends this broadcast to other KNXnet/IP routers which in turn forward it to their KNX Subnetworks and process the telegram themselves if the programming mode is active.
- 3) The tool software shall check if more than one device answers with "programming mode active" and if so shall abort procedure with a descriptive message about the reason for aborting the procedure. The KNXnet/IP router on this KNX Subnetwork forwards the answer(s) to this KNX Subnetwork.
- 4) The tool software shall send a KNX write (broadcast) to write the KNX Individual Address into the KNXnet/IP device. The KNXnet/IP router on this KNX Subnetwork sends this broadcast to other KNXnet/IP routers, which in turn forward it to their KNX Subnetworks and process the telegram themselves if the programming mode is active.
- 5) The tool software shall establish a KNX connection to the KNXnet/IP device to download the configuration parameters (properties). This requires that the KNXnet/IP router existing on the same KNX Subnetwork as the tool software shall be configured first before other KNXnet/IP devices can be configured.
- 6) After downloading the parameters the tool software shall reset the KNXnet/IP device for the parameters to become effective.

5.2.6.5.3 Configuration procedure for configuration via KNXnet/IP Device Management

The tool software is directly connected to a LAN. Two KNX Subnetworks, (1.1) and (1.2), are connected to the LAN via two KNXnet/IP routers. All devices including the routers are initially unconfigured.

The tool software shall use KNXnet/IP Device Management to configure the two KNXnet/IP routers.

This is the configuration procedure for assignment of the KNX Individual Address and configuration of parameters of a KNXnet/IP device:

- 1) The tool software shall request the user to activate the programming mode of the KNXnet/IP device.

NOTE This can be done by pressing the Programming Button on the KNXnet/IP device.

- 2) The tool software shall send a KNXnet/IP SEARCH_REQUEST frame.
- 3) The tool software shall check if more than one KNXnet/IP device (service container) answers with device status "programming mode active" and if so abort procedure.
- 4) The tool software shall use the IP address from the SEARCH_RESPONSE frame of the KNXnet/IP device to establish a device management connection to the device.
- 5) The tool software shall set the KNX Individual Address, project identifier and subnetwork information if applicable.
- 6) At this point, the tool software can download additional parameters if necessary.
- 7) After successfully disconnecting the device, management connection or sending a reset_req service to the device, the changed values shall be written and the device shall be restarted.

5.2.7 Data Packet structures

5.2.7.1 Introduction

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

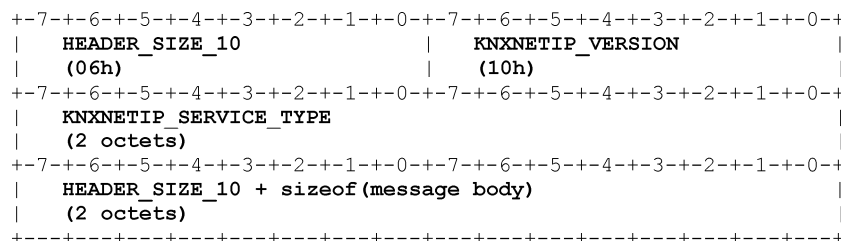


Figure 9 — KNXnet/IP message header

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

5.2.7.2 Common constants

5.2.7.2.1 HEADER_SIZE_10

This constant with value 06h shall identify the KNXnet/IP header as defined in protocol version 1.0.

5.2.7.2.2 KNXNETIP_VERSION_10

This constant with value 10h shall identify the KNXnet/IP protocol version 1.0.

5.2.7.3 Common error codes

5.2.7.3.1 E_NO_ERROR

This constant with value 00h shall identify a successful operation.

5.2.7.3.2 E_HOST_PROTOCOL_TYPE

This constant with value 01h shall identify that the requested host protocol is not supported by the KNXnet/IP device.

5.2.7.3.3 E_VERSION_NOT_SUPPORTED

This constant with value 02h shall identify that the requested protocol version is not supported by the KNXnet/IP device.

5.2.7.3.4 E_SEQUENCE_NUMBER

This constant with value 04h shall identify that the received sequence number is out of order.

5.2.7.4 KNXnet/IP services

5.2.7.4.1 Core KNXnet/IP services

5.2.7.4.1.1 SEARCH_REQUEST

This constant with value 0201h shall identify the KNXnet/IP service type sent by KNXnet/IP client to search available KNXnet/IP servers.

5.2.7.4.1.2 SEARCH_RESPONSE

This constant with value 0202h shall identify the KNXnet/IP service type sent by KNXnet/IP server when responding to a KNXnet/IP SEARCH_REQUEST.

5.2.7.4.1.3 DESCRIPTION_REQUEST

This constant with value 0203h shall identify the KNXnet/IP service type sent by KNXnet/IP client to a KNXnet/IP server to retrieve information about capabilities and supported services.

5.2.7.4.1.4 DESCRIPTION_RESPONSE

This constant with value 0204h shall identify the KNXnet/IP service type sent by KNXnet/IP server in response to a DESCRIPTION_REQUEST to provide information about the server implementation sent.

5.2.7.4.1.5 CONNECT_REQUEST

This constant with value 0205h shall identify the KNXnet/IP service type sent by KNXnet/IP client to establish a communication channel with a KNXnet/IP server.

5.2.7.4.1.6 CONNECT_RESPONSE

This constant with value 0206h shall identify the KNXnet/IP service type sent by KNXnet/IP server in response to a CONNECT_REQUEST telegram.

5.2.7.4.1.7 CONNECTIONSTATE_REQUEST

This constant with value 0207h shall identify the KNXnet/IP service type sent by KNXnet/IP client requesting the connection state of an established connection with a KNXnet/IP server.

5.2.7.4.1.8 CONNECTIONSTATE_RESPONSE

This constant with value 0208h shall identify the KNXnet/IP service type sent by KNXnet/IP server when receiving a CONNECTIONSTATE_REQUEST for an established connection.

5.2.7.4.1.9 DISCONNECT_REQUEST

This constant with value 0209h shall identify the KNXnet/IP service type sent by KNXnet/IP device, typically the client, to terminate an established connection.

5.2.7.4.1.10 DISCONNECT_RESPONSE

This constant with value 020Ah shall identify the KNXnet/IP service type sent by KNXnet/IP device, typically the server, in response to a DISCONNECT_REQUEST.

5.2.7.5 Placeholders

5.2.7.5.1 Host Protocol Address Information (HPAI)

The Host Protocol Address Information structure (HPAI) shall be the address information required to uniquely identify a communication channel on the host protocol. Its size shall vary between different host protocols. For the specific definition of the HPAI, consult the host protocol dependent addendums of the KNXnet/IPstandard.

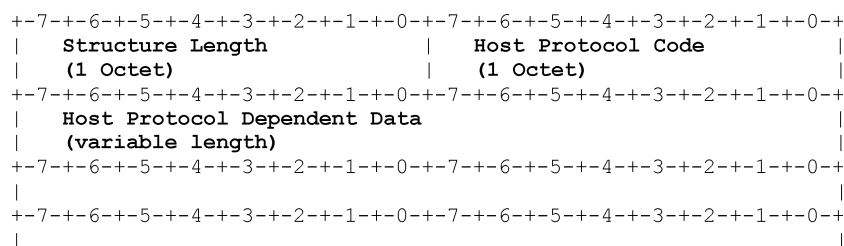


Figure 10 — HPAI structure binary format

5.2.7.5.2 Connection Request Information (CRI)

The Connection Request Information structure (CRI) shall be the additional information needed for different types of communication channels to fulfil a connection request. As this structure shall contain two substructures including host protocol independent data as well as host protocol dependent information, the specific definition of the CRI can be found in the description of the connection type with consultancy of the host protocol dependent parts of the KNXnet/IPstandard.

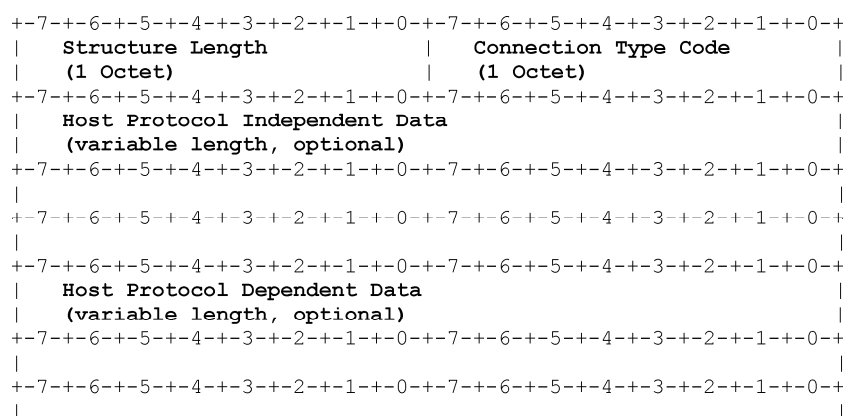


Figure 11 — CRI structure binary format

5.2.7.5.3 Connection Response Data Block (CRD)

The Connection Request Data Block structure (CRD) shall be the data block returned with the CONNECT_RESPONSE frame. As this structure shall contain two substructures including host protocol independent data as well as host protocol dependent information, the specific definition of the CRD can be found in the description of the connection type with consultancy of the host protocol dependent parts of the KNXnet/IP standard.

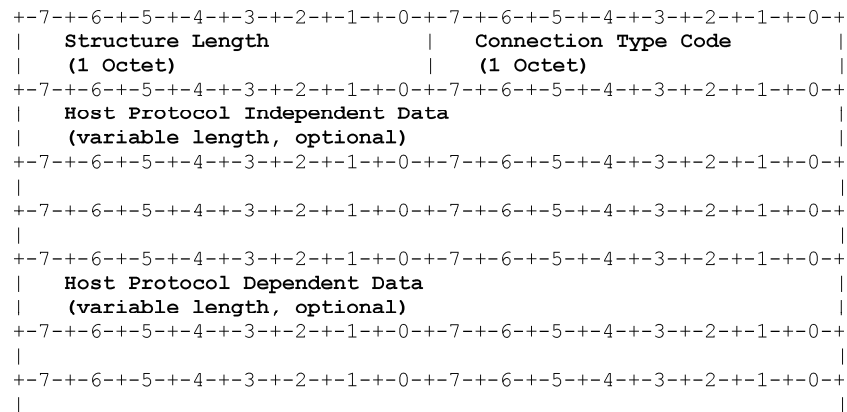


Figure 12 — CRD structure binary format

5.2.7.5.4 Description Information Block (DIB)

5.2.7.5.4.1 Use, format and general requirements

The Description Information Block structure (DIB) shall be used by a KNXnet/IP server to return a specific block of device information when responding to a DESCRIPTION_REQUEST.

At least two DIB structures shall be returned with information about the device capabilities on (1) device hardware and (2) supported service families.

More than two DIB structures may be returned in one DESCRIPTION_RESPONSE frame.

The first octet of each DIB shall contain the length of the DIB structure. The second octet shall declare the DIB structure type. Then the actual data of the DIB shall be appended. The structure shall always have an even number of octets which may have to be achieved by padding with 00h in the last octet of the DIB structure.

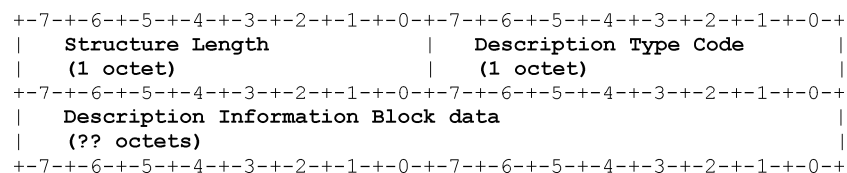


Figure 13 — Description structure binary format

Table 4 lists the valid description type codes.

Table 4 — Description type codes

Description type	Value	V.	Description
DEVICE_INFO	01h	1	Device information e.g. KNX medium
SUPP_SVC_FAMILIES	02h	1	Service families supported by the device
IP_CONFIG	03h	1	IP configuration of the device
IP_CUR_CONFIG	04h	1	Current IP configuration of the device
IP_CONFIG	05h	1	KNX addresses used by/assigned to the device
Reserved	06h – FDh	1	Reserved for future use
MFR_DATA	FEh	1	DIB structure for further data defined by device manufacturer
not used	FFh	1	Not used

5.2.7.5.4.2 Device information DIB

The device information DIB shall have the structure as given below.

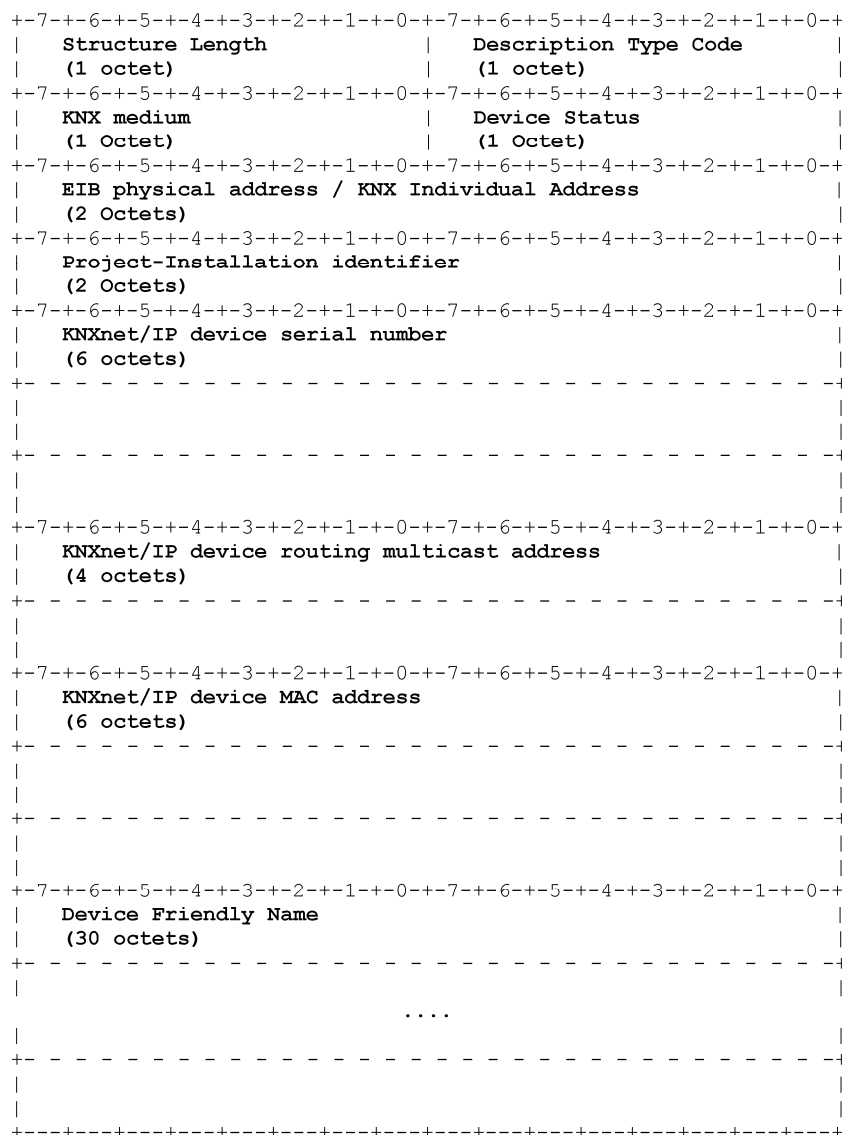


Figure 14 — Structure of device information DIB

— KNX medium

The KNX medium codes shall be as specified in Table 5 below.

Table 5 — KNX medium codes

KNX medium code	KNX medium
01h	reserved
02h	TP1
04h	PL110
08h	reserved
10h	RF
20h	KNX IP

Exactly one single bit shall be set.

— **Device Status**

The Device Status octet shall be as specified as:

Table 6

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	programming mode

— **Project-installation Identifier**

The Project-installation Identifier shall be defined as:

Table 7

bit 15 – 4	bit 3 – 0
Project number	Installation number

The Project-installation Identifier shall solely be assigned by ETS and shall be used to uniquely identify KNXnet/IP devices in a project with more than one KNX installation, i.e. more than 15 areas with 12 lines, or in an environment with more than one KNX project.

— **KNXnet/IP device KNX Serial Number**

The KNXnet/IP device serial number shall be the KNX serial number of the KNXnet/IP device. This information may be used to identify the device or set the Individual Address.

— **KNXnet/IP device routing multicast address**

The KNXnet/IP device routing multicast address shall be the multicast address used by a KNXnet/IP router for KNXnet/IP Routing. KNXnet/IP devices that do not implement KNXnet/IP Routing shall set this value to 0.0.0.0. KNX serial number of the KNXnet/IP device. This information may be used if KNXnet/IP Routing messages need to be sent to KNXnet/IP routers that do not use the default KNXnet/IP routing multicast address, which shall be equal to the KNXnet/IP system setup multicast address.

— **KNXnet/IP device MAC address**

The KNXnet/IP device MAC address is the Ethernet MAC address of the KNXnet/IP device. This information may be used to identify the device on the Ethernet to a server allocating network resources, specifically the unicast IP address for the KNXnet/IP device.

— **Device Friendly Name**

The device friendly name may be any NULL (00h) terminated ISO/IEC 8859-1 character string with a maximum length of 30 octets. This name may be used to identify the device to a user. Unused octets are filled with the NULL (00h) character.

5.2.7.5.4.3 Supported service families DIB

The supported service families DIB shall have this structure:

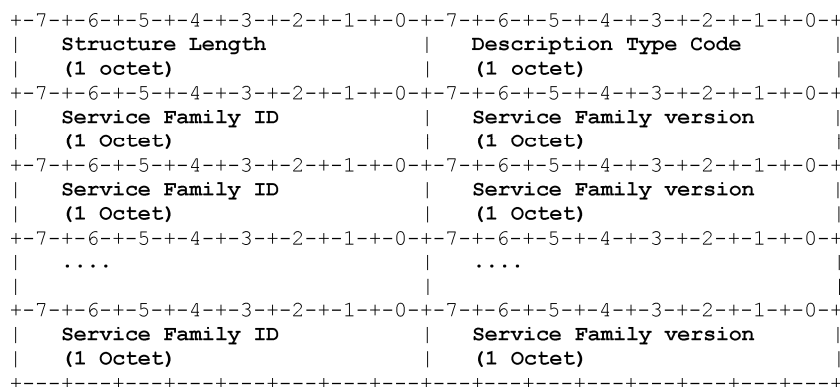


Figure 15 — Supported services families DIB

The service family ID's shall be the high octet of the service type ID. A list of service type ID's can be found in Annex A, Table A.2.

The version of a service family shall refer to the version of the corresponding KNXnet/IP chapter document. The version is only updated when the document itself is updated after it has gone through the KNX process. Any version of a service family shall be backwards compatible with previous versions, i.e. all services shall be implemented and supported. The service family version is an integer. It does not represent the manufacturer's implementation version.

5.2.7.5.4.4 Manufacturer data DIB

The manufacturer data DIB has this structure:

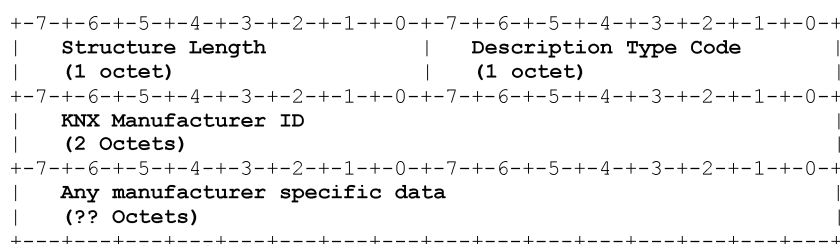


Figure 16 — Manufacturer data DIB

The KNX manufacturer ID shall be added to clearly identify the manufacturer. This information is not necessarily encoded in the KNXnet/IP serial number (6 octets).

The manufacturer data DIB may contain any manufacturer specific data.

5.2.7.6 Discovery

5.2.7.6.1 SEARCH_REQUEST

The SEARCH_REQUEST frame shall be sent by a KNXnet/IP client via multicast to the discovery endpoints of any listening KNXnet/IP server. As communication with the discovery endpoint shall be connection- and stateless, the client's discovery endpoint address information shall be included in the KNXnet/IP body.

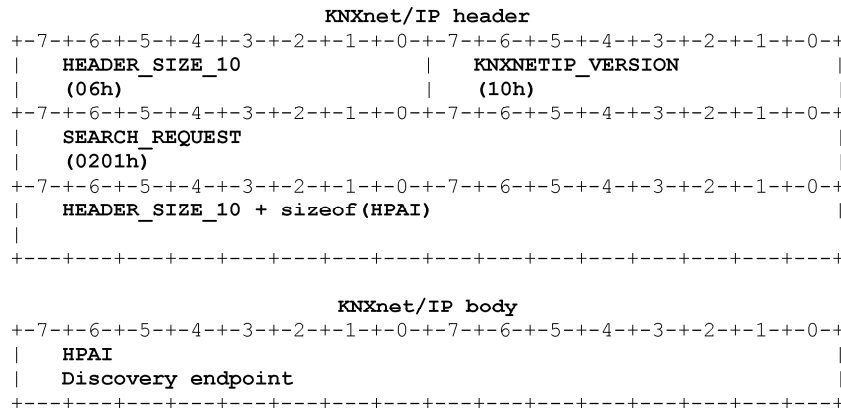


Figure 17 — SEARCH_REQUEST frame binary format

5.2.7.6.2 SEARCH_RESPONSE

The SEARCH_RESPONSE frame shall be sent by the KNXnet/IP server as an answer to a received SEARCH_REQUEST frame. It shall be addressed to the client's discovery endpoint using the HPAI included in the received frame.

The HPAI of the server's own control endpoint shall be carried in the KNXnet/IP body of the SEARCH_RESPONSE frame along with the description of the device hardware and the supported service families. If the KNXnet/IP server supports more than one KNX connection, the server shall announce each of its own control endpoints in a single SEARCH_RESPONSE frame.



Figure 18 — SEARCH_RESPONSE frame binary format

5.2.7.7 Self-description

5.2.7.7.1 DESCRIPTION_REQUEST

The DESCRIPTION_REQUEST frame shall be sent by the KNXnet/IP client to the control endpoint of the KNXnet/IP server to obtain a self-description of the KNXnet/IP server device.

The KNXnet/IP body shall contain the return address information of the client's control endpoint.

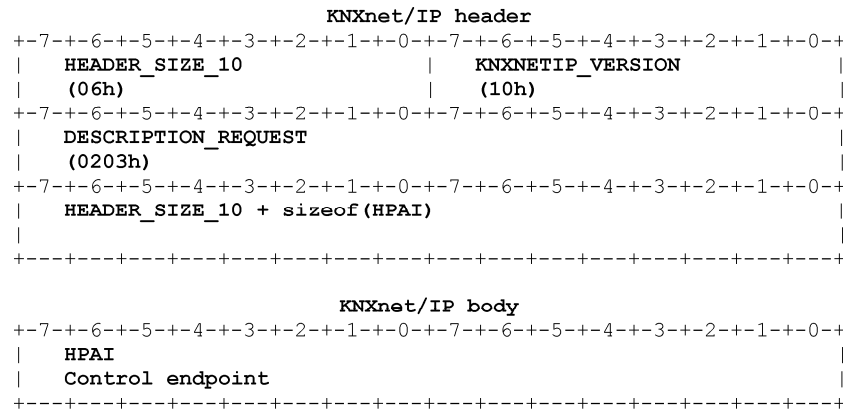


Figure 19 — DESCRIPTION_REQUEST frame binary format

The DESCRIPTION_RESPONSE frame shall be sent by the KNXnet/IP server as an answer to a received DESCRIPTION_REQUEST frame. It shall be addressed to the client's control endpoint using the HPAI included in the received frame.

The size of the KNXnet/IP body varies depending on the number of DIB structures sent by the server in response to the client's DESCRIPTION_REQUEST.

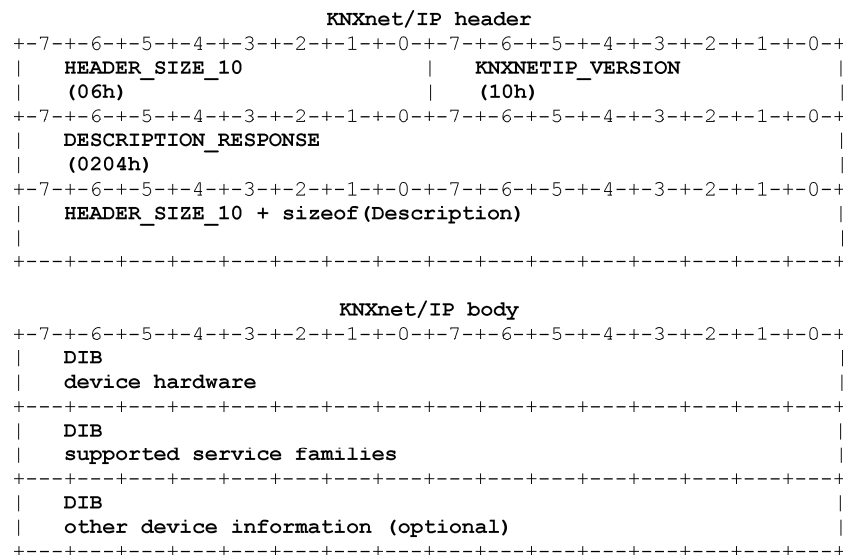


Figure 20 — DESCRIPTION_RESPONSE frame binary format

5.2.7.8 Connection management

5.2.7.8.1 CONNECT_REQUEST

The CONNECT_REQUEST frame shall be sent by the KNXnet/IP client to the control endpoint of the KNXnet/IP server. As for every request using control communication, the KNXnet/IP body shall begin with the return address information of the client's control endpoint.

Next follows the CRI, a variable data structure that shall include all additional information that is specific to the requested connection type (and to the underlying host protocol). The exact definition of this structure can be found in the description of the specific connection type.

Table 8 — Connection types

Connection type	Value	V.	Description
DEVICE_MGM_CONNECTION	03h	1	Data connection used to configure a KNXnet/IP device
TUNNEL_CONNECTION	04h	1	Data connection used to forward EIB telegrams between two KNXnet/IP devices
REMLOG_CONNECTION	06h	1	Data connection used for configuration and data transfer with a remote logging server
REMCNF_CONNECTION	07h	1	Data connection used for data transfer with a remote configuration server
OBJSVR_CONNECTION	08h	1	Data connection used for configuration and data transfer with an Object Server in a KNXnet/IP device

Inside the CRI one octet shall determine the type of communication channel requested by this frame and two octets are reserved for the options for this channel. Additional KNXnet/IP documents may define more connection types and additional connection type specific connection options.

The host protocol address information of the client's data endpoint meant for the requested data connection shall complete the body of the CONNECT_REQUEST frame.



Figure 21 — CONNECT_REQUEST frame binary format

5.2.7.8.2 CONNECT_RESPONSE

The CONNECT_RESPONSE frame shall be sent by the KNXnet/IP server as an answer to a received CONNECT_REQUEST frame. It shall be addressed to the client's control endpoint using the HPAI included in the received frame.

The size of the KNXnet/IP body varies according to the success or failure of the client's CONNECT_REQUEST.

If the connection request could be successfully fulfilled with all the requested options, the body shall contain a communication channel ID that uniquely identifies this connection with the KNXnet/IP server. The communication channel ID shall be the first octet of the body.

The second octet of the body shall contain the status information of the connection request. This status information can contain error information regarding the request itself or regarding the connection type specific information.

Table 9 — Common CONNECT_RESPONSE status codes

Error constant	Value	V.	Description
E_NO_ERROR	00h	1	The connection was established successfully
E_CONNECTION_TYPE	22h	1	The requested connection type is not supported by the KNXnet/IP server device
E_CONNECTION_OPTION	23h	1	One or more requested connection options are not supported by the KNXnet/IP server device
E_NO_MORE_CONNECTIONS	24h	1	The KNXnet/IP server device could not accept the new data connection because its maximum amount of concurrent connections are already busy

The Host Protocol Address Information of the server's data endpoint prepared for this data connection shall be the next block of data in the body of a successful CONNECT_RESPONSE frame.

The Connection Response Data Block containing connection type specific response data shall complete the body of a successful CONNECT_RESPONSE frame.

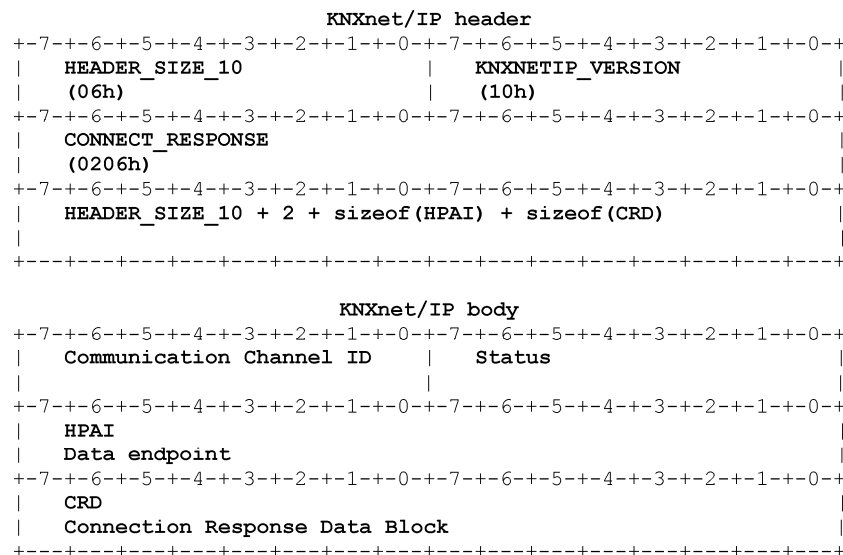


Figure 22 — CONNECT_RESPONSE frame binary format

5.2.7.8.3 CONNECTIONSTATE_REQUEST

The CONNECTIONSTATE_REQUEST frame shall be sent by the KNXnet/IP client to the control endpoint of the KNXnet/IP server. The first octet of the KNXnet/IP body shall contain the communication channel ID that the KNXnet/IP server uses to uniquely identify the data connection for this connection state request. The second octet shall be reserved for future use.

The HPAI with the return address information of the client's control endpoint shall be added after the communication channel ID.

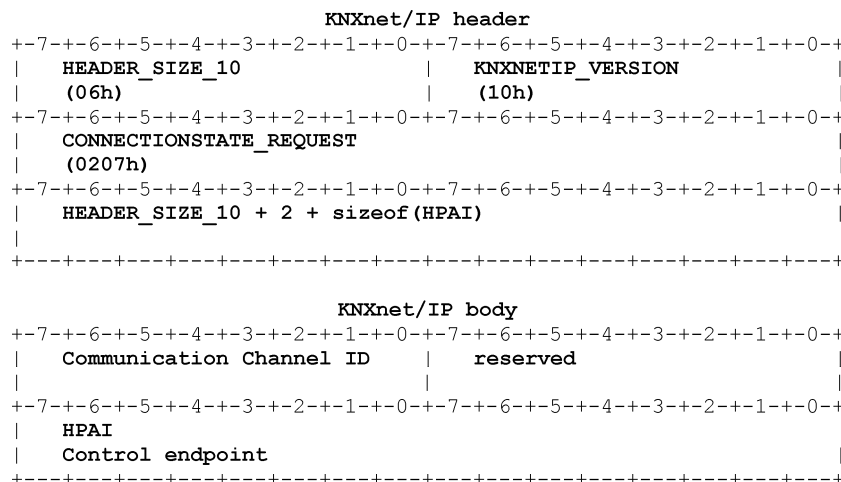


Figure 23 — CONNECTIONSTATE_REQUEST frame binary format

5.2.7.8.4 CONNECTIONSTATE_RESPONSE

The CONNECTIONSTATE_RESPONSE frame shall be sent by the KNXnet/IP server as an answer to a received CONNECTIONSTATE_REQUEST frame. It shall be addressed to the client's control endpoint using the HPAI included in the received frame.

The first octet of the KNXnet/IP body shall contain the communication channel ID that the KNXnet/IP client passed to the KNXnet/IP with the CONNECTIONSTATE_REQUEST frame.

The second octet of the KNXnet/IP body shall contain the status information of the connection state request. The following table lists the status codes that are defined for the CONNECTIONSTATE_RESPONSE frame:

Table 10 — CONNECTIONSTATE_RESPONSE status codes

Error constant	Value	V.	Description
E_NO_ERROR	00h	1	The connection state is normal
E_CONNECTION_ID	21h	1	The KNXnet/IP server device could not find an active data connection with the specified ID
E_DATA_CONNECTION	26h	1	The KNXnet/IP server device detected an error concerning the data connection with the specified ID
E_KNX_CONNECTION	27h	1	The KNXnet/IP server device detected an error concerning the EIB bus/KNX subsystem connection with the specified ID

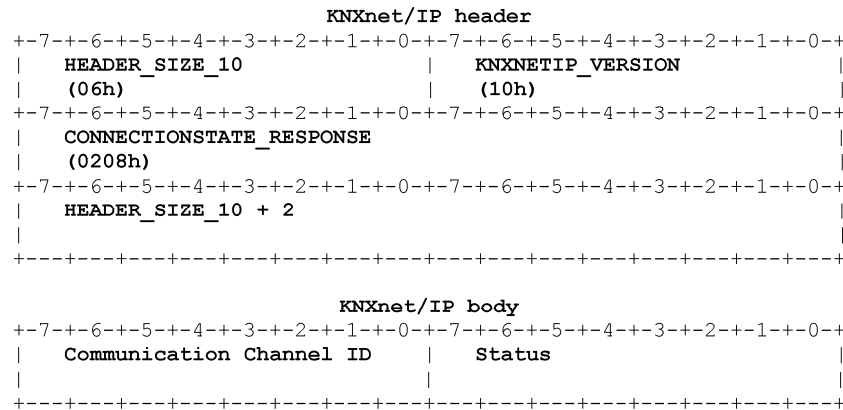


Figure 24 — CONNECTIONSTATE_RESPONSE frame binary format

5.2.7.8.5 DISCONNECT_REQUEST

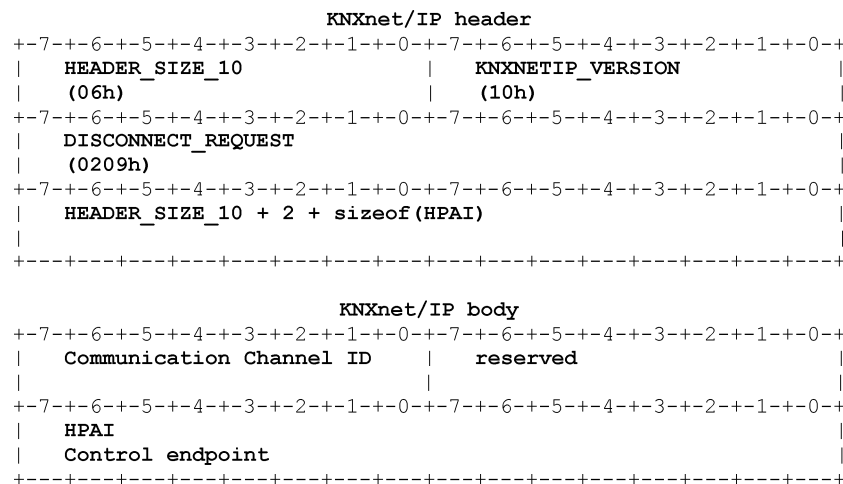


Figure 25 — DISCONNECT_REQUEST frame binary format

5.2.7.8.6 DISCONNECT_RESPONSE

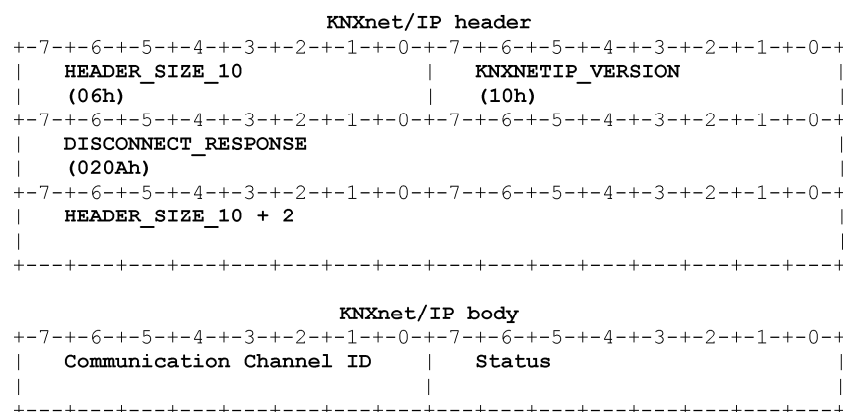


Figure 26 — DISCONNECT_RESPONSE frame binary format

5.2.8 IP Networks

5.2.8.1 Introduction

The KNXnet/IP protocol is used to tunnel or route KNX data over the widely spread Internet Protocol (IP), enabling remote access and maintenance across long distances, as well as usage as high-speed backbone for KNX networks.

This part of the standard defines which IP parameters and features are supported by KNXnet/IP.

It is assumed that the reader is familiar with the Internet Protocols TCP and UDP.

5.2.8.2 Physical vs. logical network

IP networks are not like KNX networks: KNX networks are physical busses by nature. This implies that all devices on the channel will by default receive all packets transmitted on the network. In addition, when a new device is added to the network it is not necessary that other devices on the network become aware of it before they can exchange packets. To transmit a telegram over KNX, it is only necessary that a device be capable of physically transmitting the telegram on the bus, nothing more. If a device is simply physically connected to a KNX network, it is capable of exchanging telegrams with other devices on the channel.

By contrast, an IP network is not physical, but logical in nature. There are a number of different physical media that can support IP communications and any of them should be capable of supporting tunnelling KNXnet/IP frames. Because it is dealt with a logical channel, it is necessary to “construct” the channel by informing each device on the channel of the existence of the other devices on that channel. In other words before a device can transmit a packet to some other device on an IP channel it shall be made aware of how to specifically send a packet to that device, i.e. its IP address.

Another significant difference between physical and logical networks is that in the case of typical physical networks it is possible to calculate fixed upper bounds on the length of time it will take a packet to traverse from one device to another once the packet is transmitted on the channel. This is not always possible for IP networks. The deviation of packet delivery times between KNXnet/IP devices on an IP channel are much higher than those experienced with native KNX devices.

The IP channel is used as an intermediary transport mechanism for the KNX telegrams by a variety of KNXnet/IP devices. When a KNXnet/IP packet is transported on an IP channel, an IP message encapsulating the KNX telegram is sent to other KNXnet/IP devices on that IP channel. The IP channel is specified by the list of unicast IP addresses, exactly one for each KNXnet/IP device. There is no maximum to the number of KNXnet/IP devices on a single IP network.

5.2.8.3 Transport mechanisms

IP is a network level protocol. It is designed to operate over a wide range of physical media and link layer protocols. As such, this document does not specify anything about the link or physical layers of the IP stack.

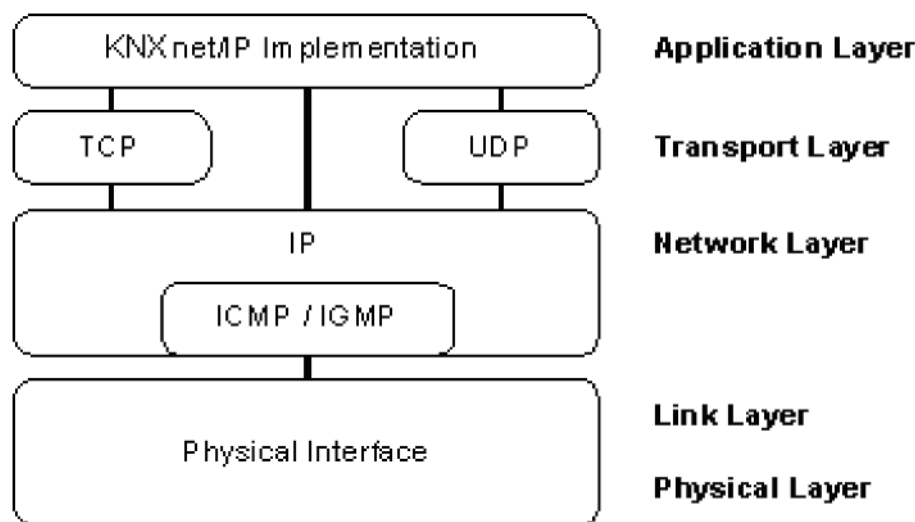


Figure 27 — IP protocol stack

Three most common mechanisms used to transport IP packets are raw IP, TCP and UDP. TCP and UDP are transport protocols built on top of IP.

TCP implements a reliable, connection-oriented end-to-end transport service. It includes provisions to guarantee the correct transmission and to preserve the ordering of the received data stream.

UDP on the other hand implements a best effort datagram service. Raw IP is a mechanism implemented by a number of operating systems to provide host applications with direct access to the IP layer, bypassing any transport service.

Although it is possible to use any of these protocols as a KNXnet/IP host protocol, from an application point of view it is much easier to simply exclude raw IP and restrict the specification to TCP and UDP.

5.2.8.4 UDP and TCP

Since the KNX protocol itself employs end-to-end acknowledgment, it is not necessary to guarantee tunnelling IP packets containing KNX data telegrams are received between the KNXnet/IP devices. Using a reliable transport service introduces unnecessary protocol overhead in this case, which makes TCP less efficient than UDP or raw IP. On the other hand, there will be configuration and status messages exchanged between KNXnet/IP devices that do not contain tunnelled KNX data and yet need to be received reliably.

TCP has the advantages of reliable delivery service and hence will guarantee that the received packet ordering is preserved. On the down side, it does not support multi-cast addressing and is less efficient than UDP. TCP also consumes more resources of the KNXnet/IP device to implement than UDP. UDP is more efficient in carrying tunnelled KNX data messages, but for the lack of a reliable delivery, service will not guarantee that the packet ordering is preserved.

Given the increased efficiencies of UDP regarding the transport of KNX data messages and its support of multi-cast addressing, it will be used as the default to communicate between KNXnet/IP devices. All KNXnet/IP devices shall support UDP. The reliability advantages of TCP may be supported in addition to UDP. TCP support in KNXnet/IP devices is OPTIONAL.

To address the sequencing issue there shall be the sequence counter option added to the connection header to help in sequencing.

Using UDP, datagrams can be sent using either unicast or multi-cast addressing. Unicast is point to point meaning that a datagram is sent from one IP host to a single other IP host. When sending the same datagram to multiple IP hosts as it is necessary for routing of KNX data, it is much more efficient to use multi-cast addressing. It is therefore recommended that KNXnet/IP devices support both unicast and multi-cast IP

addressing although it is not required that a KNXnet/IP device supports multi-casts in order to inter-operate with a KNXnet/IP device that does.

5.2.8.5 IP Address Assignment

5.2.8.5.1 IP unicast address

5.2.8.5.1.1 Fixed IP address

A fixed IP address is assigned to the device through a user interface, via ETS or some other tool. This IP address assignment is fixed.

Any KNXnet/IP device shall support fixed assigned IP addresses.

5.2.8.5.1.2 BootP / DHCP

A BootP or DHCP server is designed to automatically assign an IP address to a device. Configuration of both types of servers is part of network administration and is available on all network server platforms (Windows NT Server, Windows 2000 Server, Windows XP Server, Unix, Linux). Pre-administered DHCP servers like DSL modems or ISDN routers can also be used.

Either a BootP or a DHCP client shall be implemented on a KNXnet/IP device.

5.2.8.5.1.3 AutoIP

A device implementing AutoIP is capable of assigning itself a unicast IP address in the range of 169.254.1.0 to 169.254.254.255.

A KNXnet/IP device may implement AutoIP.

5.2.8.5.1.4 IP address assignment procedure

Table 11 describes the address assignment procedure for KNXnet/IP devices.

Table 11 — Address assignment procedure

1	IF	Fixed IP address assigned to KNXnet/IP device	THEN	→ 2
			ELSE	→ 6
2	IF	BootP or DHCP address assignment is enabled	THEN	→ 3
			ELSE	Use fixed IP address already assigned
3	IF	BootP or DHCP address assignment is successful	THEN	Use newly assigned IP address
			ELSE	→ 4
4	IF	AutoIP address assignment is enabled	THEN	→ 5
			ELSE	Use fixed IP address already assigned
5	IF	AutoIP address assignment is successful	THEN	Use newly assigned IP address
			ELSE	Use fixed IP address already assigned
6	IF	BootP or DHCP address assignment is enabled	THEN	→ 7
			ELSE	→ 8
7	IF	BootP or DHCP address assignment is successful	THEN	Use newly assigned IP address
			ELSE	→ 8
8	IF	AutoIP address assignment is enabled	THEN	→ 9
			ELSE	No IP address is assigned. Enter 0.0.0.0 in IP address field of HPAI
9	IF	AutoIP address assignment is successful	THEN	Use newly assigned IP address.
			ELSE	No IP address is assigned. Enter 0.0.0.0 in IP address field of HPAI

5.2.8.5.2 IP multicast addresses

5.2.8.5.2.1 KNXnet/IP system setup multicast address

To ensure that any KNXnet/IP device can be reached by the KNXnet/IP Core discovery services a "System Setup Multicast Address" is defined. The value of the "System Setup Multicast Address" shall be 224.0.23.12.

5.2.8.5.2.2 KNXnet/IP routing multicast address

Any KNXnet/IP device implementing KNXnet/IP routing shall have a "Routing Multicast Address". This address shall be derived from the "System Setup Multicast Address" by adding an offset. This offset has a default value of zero. If there is more than one installation in a project KNXnet/IP routers in separate installations shall be assigned to different Routing Multicast Addresses.

5.2.8.6 KNXnet/IP host protocol

5.2.8.6.1 Device specification

A KNXnet/IP device shall behave like any standard IP host capable of exchanging IP packets with any other IP host either on the same IP subnet or anywhere else in the IP network cloud. A KNXnet/IP device shall have a single unicast IP address, shall belong to at least one IP multi-cast group, and may be capable of belonging to up to 16 multi-cast groups. A KNXnet/IP device shall support multi-casting.

A KNXnet/IP device shall support these IP protocols: ARP, ICMP, IGMP, BootP/DHCP⁴ and UDP. Additionally it may support other IP protocols like RARP, TCP, NTP, FTP, HTTP, SMTP, DNS or SNMP.

In order to initiate communication via TCP/UDP to a KNXnet/IP device it is necessary to define a fixed port number for the discovery procedure in addition to the individual IP address.

To support the routing of IP packets between subnets or through the Internet KNXnet/IP devices shall be compatible with whatever standard mechanisms (IP routers, switches, etc.) are required to perform the IP routing functions.

5.2.8.6.2 Host Protocol Address Information

The Host Protocol Address Information shall contain the information that is necessary to uniquely identify an Internet Protocol transport connection endpoint. This shall include the network layer address and the transport layer identifier called Port number. Both, IP address and port number, are stored binary in network octet order.

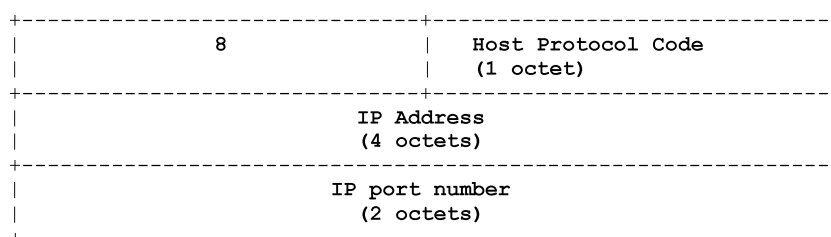


Figure 28 — IP Host Protocol Address Information binary format

Table 12 — Host protocol codes for IP network

Constant name	Value	V.	Description
IPV4_UDP	01h	1	Identifies an Internet Protocol version 4 address and port number for UDP communication
IPV4_TCP	02h	1	Identifies an Internet Protocol version 4 address and port number for TCP communication

5.2.8.6.3 KNXnet/IP Endpoints

5.2.8.6.3.1 General

The use of the KNXnet/IP endpoints as a logical view of the communication between KNXnet/IP devices results in a big flexibility of the actual implementation using the specific host protocol.

As mentioned above KNXnet/IP devices can support UDP or optionally TCP as the transport layer protocol for IP communication. These IP channels can dynamically be negotiated between the KNXnet/IP devices using the HPAI structure of the KNXnet/IP frames. Because of the point-to-multipoint requirement of the discovery communication, UDP at port 3671 is the only allowed transport mechanism.

The KNXnet/IP port number 3671 shall be used for the discovery end point. It may be used for the data and control end points too. This is not mandatory. The KNXnet/IP server port number returned in response to a CONNECT_REQUEST may be any valid port number.

4) BootP/DHCP: It is essential that either one is implemented.

It is therefore possible to implement a KNXnet/IP device that uses only one bi-directional UDP port for all communication or one could implement a KNXnet/IP device that uses two unidirectional UDP ports for the discovery endpoint and for each control endpoint and TCP ports for data connections.

5.2.8.6.3.2 Discovery Endpoint

Only UDP is allowed for discovery endpoint communication. An attempt to request TCP communication will result in a host protocol type error.

The KNXnet/IP client shall send a SEARCH_REQUEST frame using UDP local broadcast from any source port to the fixed discovery endpoint destination at port 3671. Any KNXnet/IP server receiving this request shall ignore the source information at IP level. Only the HPAI of the received KNXnet/IP frame is relevant. The KNXnet/IP servers shall then send their SEARCH_RESPONSE frame(s) using UDP unicast from any source port to the requested destination.

5.2.8.6.3.3 Control Endpoint

UDP and TCP are allowed for control endpoint communication. Every KNXnet/IP device shall support UDP communication. TCP communication is OPTIONAL.

The KNXnet/IP client shall receive the port information about the KNXnet/IP server's control endpoint from the HPAI of the SEARCH_RESPONSE frames. A KNXnet/IPserver shall not change this port once it is announced.

The KNXnet/IP server shall receive the complete address information about the KNXnet/IP client's control endpoint with every new control request. Although possible, it is recommended that the client does not change this port either.

5.2.8.6.3.4 Data endpoints

UDP and TCP are allowed for data endpoint communication. Every KNXnet/IP device shall support UDP communication. TCP communication is OPTIONAL. Data endpoints can be connection oriented for point-to-point communication or connectionless for point-to-multipoint communication as it is necessary for KNXnet/IP routing. All connectionless data endpoints use special Host Protocol Address Information structures for KNXnet/IP multicasts. These IP addresses and port numbers are fixed and defined in the corresponding KNXnet/IP service protocol description.

5.2.8.6.3.5 Network Address Translation (NAT)

If KNXnet/IP communication has to traverse across network routers using Network Address Translation (NAT) the KNXnet/IP client shall set the value of the IP address and/or the port number in the HPAI to zero to indicate NAT traversal to the receiving KNXnet/IP server.

For the IP address and port number in the datagrams sent from the KNXnet/IP server to the KNXnet/IP client, the KNXnet/IP server shall replace the zero value for the IP address and/or the port number in the HPAI by the corresponding IP address and/or port number in the IP package received and use this value as the target IP address or port number for the response to the KNXnet/IP client.

Typically the KNXnet/IP client should set both the IP address and the port number to zero but may choose to set only one (IP address or port number) to zero if required by the application and possible in the given network configuration.

5.2.9 Certification

5.2.9.1 Introduction

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

5.2.9.2 Supported services

The supported services for KNXnet/IP and the Internet Protocol (IP) are listed in Tables 13 and 14. IPv6 will be specified in the future and is listed for completeness only.

Table 13

Service name	Sent from... to...	Implementation is
SEARCH_REQUEST	Client → Server	M
SEARCH_RESPONSE	Server → Client	M
DESCRIPTION_REQUEST	Client → Server	M
DESCRIPTION_RESPONSE	Server → Client	M
CONNECT_REQUEST	Client → Server	M
CONNECT_RESPONSE	Server → Client	M
CONNECTIONSTATE_REQUEST	Client → Server	M
CONNECTIONSTATE_RESPONSE	Server → Client	M
DISCONNECT_REQUEST	Client → Server Server → Client	M
DISCONNECT_RESPONSE	Client → Server Server → Client	M

Table 14

Service name	Client	Server
IPV4_UDP	M	M
IPV4_TCP	O	O
IPV6_UDP	to be defined	to be defined
IPV6_TCP	to be defined	to be defined

5.3 Clause 3: Device Management Specification

5.3.1 Scope

This clause “Device Management” of the KNXnet/IP standard 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.

5.3.2 KNXnet/IP Device Management

5.3.2.1 Introduction

KNX devices connected to an IP network through KNXnet/IP devices are configured by ETS as described in 5.4.

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

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