System Specifications

KNXnet/IP

Core

3

8

2

Summary

This document provides the Core KNXnet/IP specification.

Version 01.05.01 is a KNX Approved Standard.

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

Document updates

Version	Date	Modifications
1.0 DP	2004.01.07	Draft Proposal provided for Release for Voting
1.1 DV	2005.05.27	Preparation of the Draft for Voting.
1.3 AS	2008.07.02	Publication of the Approved Standard.
1.4 AS	2008.09.05	• AN106 "Phasing out TP0" integrated • AN109 "Phasing out PL132" integrated
1.4 AS	2009.06.29	• Preparation for inclusion in the KNX Specifications v2.0: editorial update.
1.4.01 AS	2011.01.04	• AN115 “Mask 5705h” – added “KNX IP” as medium to Table 2.
1.5.00	2010.06.10	• AN123 “KNXnet/IP Remote Diagnosis and Configuration” integrated.
01.05.01	2013.10.28	Editorial updates for the publication of KNX Specifications 2.1.

References

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

[01] Chapter 3/5/1 “Resources”
[02] Chapter 3/6/3 “External Message Interface”
[03] Chapter 3/7/2 “Datapoint Types”
[04] Chapter 3/8/1 “Overview” (KNXnet/IP)

Filename:	03_08_02 Core v01.05.01 AS.docx
Version:	01.05.01
Status:	Approved Standard
Savedate:	2013.10.28
Number of pages:	49

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

Content

1 Introduction

1.1 Scope

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

An overview of KNXnet/IP is presented in [04].

This Chapter 3/8/2 "Core" of the KNXnet/IP specification provides a general host protocol-independent framework that accommodates several specialized "Service Protocols" in a modular and extendible fashion.

This specification 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
- configuring, establishing and maintaining a communication channel between a KNXnet/IP Client and a KNXnet/IP Server.

This document defines a standard protocol that is implemented within KNX devices and the Engineering Tool Software (ETS) to support KNX data exchange over IP networks or other systems.

1.2 Definitions, acronyms and abbreviations

Refer to [04] for a list of definitions for the KNXnet/IP specification.

2 KNXnet/IP frames

2.1 General definitions

2.1.1 Data format

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

The data structure specifications are always noted in binary format.

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.

2.1.3 Structures

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

2.2 Frame format

The communication between KNXnet/IP devices shall be based on KNXnet/IP frames. A KNXnet/IP frame shall be 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.

```
+------------------------------+
|   KNXnet/IP Frame            |
|   (Fixed Length)             |
|                              |
+------------------------------+
|   KNXnet/IP Frame            |
|   (Variable Length)          |
|   Optional                   |
+------------------------------+
```

Figure 1 – 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.

2.3 Header

2.3.1 Description

Every KNXnet/IP frame, without any exception, shall consist of at least the common KNXnet/IP header that shall contain 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.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Header Length               |   Protocol Version            |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Service Type Identifier                                     |
|   (2 Octet)                                                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Total Length                                                |
|   (2 Octet)                                                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
```

Figure 2 – KNXnet/IP header binary format

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.

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).

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.

2.3.5 Total length

The total length shall express the total KNXnet/IP frame length in octets. The length shall include the complete KNXnet/IP frame, starting with the header length of the KNXnet/IP header and including the whole KNXnet/IP body.

3 Host protocol independence

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.

3.2 Endpoints

KNXnet/IP defines the Host Protocol Address Information (HPAI) structure, which shall be the combination of IP address and port number. The HPAI shall be the data required to send a KNXnet/IP frame to another device. The KNXnet/IP specification 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, bidirectional and connectionless endpoint for discovery if the host protocol requires discovery services. It shall support at least one bidirectional and connectionless endpoint for controlling and at least one bidirectional and connection oriented endpoint for service type related data transmission.

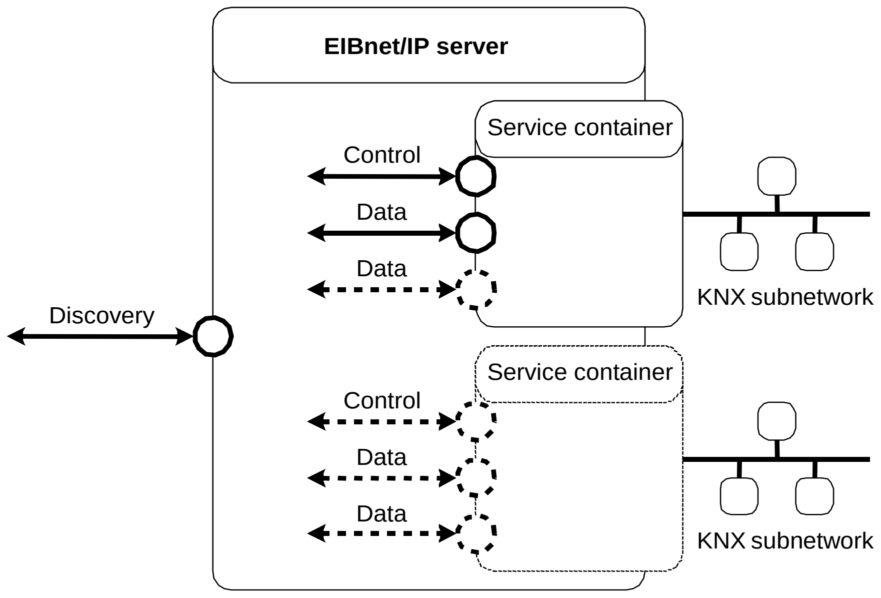

Figure 3 – KNXnet/IP Server endpoints sample configuration

The control endpoint shall uniquely address 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 connections, it is required that every KNX Subnetwork shall be represented by a different control endpoint. The KNXnet/IP Client shall therefore consider every service container represented by a control endpoint as one independent entity no matter if they are implemented in only one or two separate KNXnet/IP Server devices.

These KNXnet/IP endpoints shall 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 bidirectional 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 HPAIs.

4 Discovery and self description

4.1 Introduction

Especially for networks supporting hot-plug and where probably even the address assignment takes 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-standardized 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.

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 shall consist of a SEARCH_REQUEST data packet, sent via multicast using the discovery endpoint, which shall contain 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.

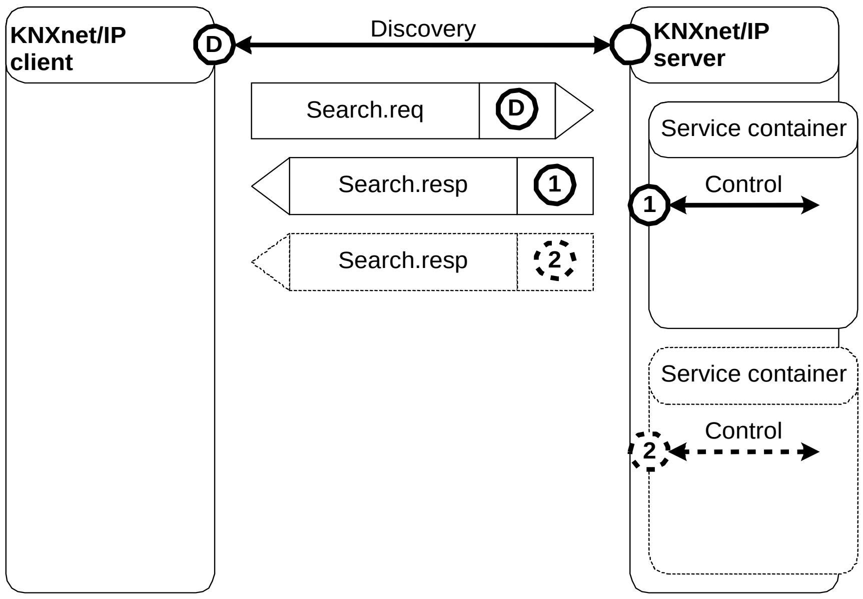

Figure 4 – Discovery procedure

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 frame to the given HPAI using its discovery endpoint. Such a response shall contain 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, however it shall 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.

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 frame 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 Communication Channels

5.1 Introduction

A communication channel shall be the data endpoint connection between a KNXnet/IP Client and a KNXnet/IP Server. Data endpoint connections shall be established for services requiring point-to-point communication e.g. KNXnet/IP 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 KNXnet/IP Tunnelling connection on the same physical access to a KNX Subnetwork in different modes – Normal Mode or Busmonitor Mode).

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 shall provide information on the requested connection type (e.g. data tunnelling or remote logging), general and connection type specific options (e.g. Data Link Layer or Busmonitor Mode [2]) 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 (with extended status information if applicable). If the request could be accepted by the server, the CONNECT_RESPONSE frame shall additionally contain an identifier as well as the HPAI of the data endpoint that the server now prepared for this communication channel [3].

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

The current protocol specification assumes that a connection shall not be shared by multiple clients. Therefore a KNXnet/IP Server shall 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.

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

[3] Note that an KNXnet/IP connection may consist due to technical reasons of multiple logical connections at the host protocol layer.

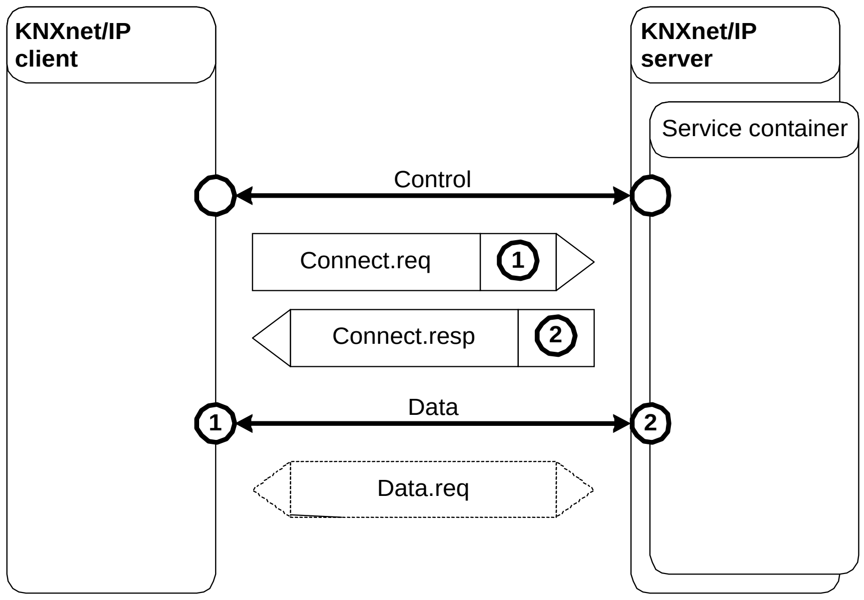

Figure 5 – Establishing a data connection

5.3 Connection Header

5.3.1 Description

The body of every KNXnet/IP frame sent over an established communication channel shall start with data fields that shall contain additional general information about the data connection. The amount of this data and what type of information is included there in particular shall be 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.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Communication Channel ID    |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Sequence Counter            |   service type specific       |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Connection Type Specific Header                             |
|   (variable length, optional)                                 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 6 – Common connection header

5.3.2 Structure length

Structure Length shall be the total length of the connection header.

5.3.3 Communication Channel ID

The KNXnet/IP Server shall assign a Communication Channel ID to each established communication channel.

5.3.4 Sequence Counter

The sequence counter shall be 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 shall 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 shall have a sequence counter of 0.

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.

5.4 Heartbeat monitoring

Host protocols not providing mechanisms for life time 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 KNXnet/IP Servers.

The client shall send a CONNECTIONSTATE_REQUEST frame regularly, i.e. every 60 seconds, 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 seconds) or the status of a received heartbeat response signalled any kind of error condition, the client shall repeat the CONNECTIONSTATE_REQUEST three (3) 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 seconds 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.

5.5 Disconnecting

Typically the client terminates the connection. During normal operation, the client shall send a DISCONNECT_REQUEST frame 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 shall signal the final termination of a previously established communication channel.

6 General implementation guidelines

6.1 Introduction

This clause defines programming guidelines that must be taken into account when implementing KNXnet/IP Servers or clients, respectively. The compliance to these guidelines is a requirement for the certification of the protocol implementation.

6.2 KNXnet/IP Servers

- If a server receives a data packet with an unsupported version field, it shall reply with a negative confirmation frame 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.

6.3 KNXnet/IP Clients

- If a client receives a data packet with an unsupported version field, it shall reply with a negative confirmation frame 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.

6.4 KNXnet/IP Router settings

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 standardized factory default configuration.

- Routers shall be shipped with a default Individual Address of FF00h.
- The Routing multicast address shall equal 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.
- KNXnet/IP Routing shall already work and KNXnet/IP 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.

6.4.2 KNXnet/IP Router IP address assignment

KNXnet/IP Routers shall support plug and play KNXnet/IP Routing out of the box even if a valid unicast IP address is 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 is not 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 is 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.

6.5 Initial setup procedures for KNXnet/IP Servers

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 shall be 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.

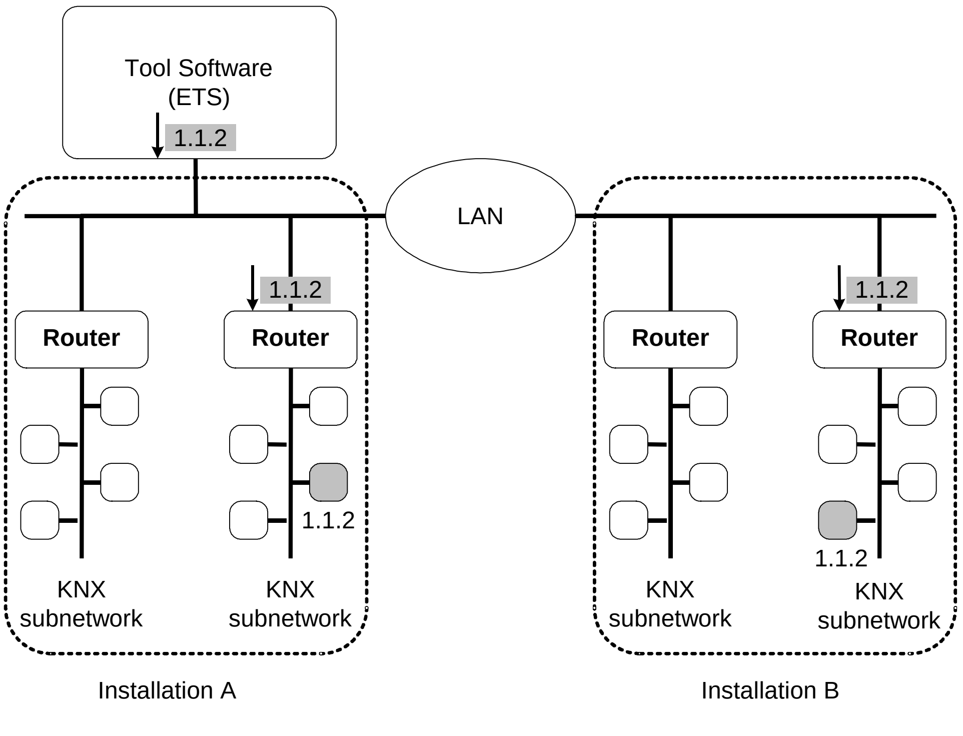

Figure 7 – KNX project with multiple installations

Under the precondition that only one installation is connected to the IP network these rules apply.

1. Upon power-up the KNXnet/IP device shall step through the procedure described in KNXnet/IP Core, clause 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 KNXnet/IP 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 KNXnet/IP Routers are not reachable. If the tool software is connected to the IP network, the configuration procedure for KNXnet/IP devices (specified below) shall apply. Before setting up KNX field media devices the tool software must firstly configure all KNXnet/IP Router devices.

8. Parameter download

   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 shall apply for KNXnet/IP devices as well as other KNX devices. If the tool software is connected to the IP network, it shall establish 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 an 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.

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 KNXnet/IP 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:

a) The tool software requests the user to activate the Programming Mode of the KNXnet/IP device.

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

b) 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.

c) 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 shall forward the answer(s) to this KNX Subnetwork.

d) 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 shall send this broadcast to other KNXnet/IP Routers, which in turn shall forward it to their KNX Subnetworks and process the telegram themselves if their Programming Mode is active.

e) The tool software shall establish a KNX Transport Layer connection to the KNXnet/IP device to download the configuration parameters (Properties).

   NOTE This requires that the KNXnet/IP Router existing on the same KNX Subnetwork as the tool software shall be configured firstly before other KNXnet/IP devices can be configured.

f) After downloading the parameters the tool software shall reset the KNXnet/IP device for the parameters to become effective.

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 KNXnet/IP 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.

a) The tool software shall requests the user to activate the Programming Mode of the KNXnet/IP device.

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

b) The tool software shall send a KNXnet/IP SEARCH_REQUEST frame.

c) The tool software shall check if more than one KNXnet/IP device (service container) answers with device status “button pressed” and if so abort the procedure.

d) The tool software shall use the IP address from the SEARCH_RESPONSE frame of the KNXnet/IP device to establish a KNXnet/IP device management connection to the device.

e) The tool software shall set the KNX Individual Address, project identifier and Subnetwork information if applicable.

f) At this point the tool software can download additional parameters if necessary.

g) After successfully disconnecting the KNXnet/IP device management connection or sending a reset_req service to the device, the changed values shall be written and the device shall be restarted.

7 Frame structures

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.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNXNETIP_SERVICE_TYPE                                       |
|   (2 octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(message body)                       |
|   (2 octets)                                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 8 – KNXnet/IP frame header

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

7.2 Common constants

7.2.1 HEADER_SIZE_10

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

7.2.2 KNXNETIP_VERSION_10

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

7.3 Common error codes

7.3.1 E_NO_ERROR

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

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.

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.

7.3.4 E_SEQUENCE_NUMBER

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

7.4 KNXnet/IP services

7.4.1 Core KNXnet/IP services

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.

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.

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.

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 KNXnet/IP Server implementation.

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.

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 frame.

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.

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.

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 KNXnet/IP client, to terminate an established connection.

7.4.1.10 DISCONNECT_RESPONSE

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

7.5 Placeholders

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/IP specification.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Host Protocol Code          |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Host Protocol Dependent Data                                |
|   (variable length)                                           |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
```

Figure 9 – HPAI structure binary format

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/IP specification.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Connection Type Code        |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Host Protocol Independent Data                              |
|   (variable length, optional)                                 |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Host Protocol Dependent Data                                |
|   (variable length, optional)                                 |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
```

Figure 10 – CRI structure binary format

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 specification.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Connection Type Code        |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Host Protocol Independent Data                              |
|   (variable length, optional)                                 |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Host Protocol Dependent Data                                |
|   (variable length, optional)                                 |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|                                                               |
```

Figure 11 – CRD structure binary format

7.5.4 Description Information Block (DIB)

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.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Description Information Block data                          |
|   (?? octets)                                                 |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
```

Figure 12 – Description structure binary format

Table 1 lists the valid description type codes.

Table 1 - Description type codes

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

7.5.4.2 Device information DIB

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

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNX medium                  |   Device Status               |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNX Individual Address                                      |
|   (2 Octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Project-Installation identifier                             |
|   (2 Octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNXnet/IP device KNX Serial Number                          |
|   (6 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNXnet/IP device routing multicast address                  |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNXnet/IP device MAC address                                |
|   (6 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Device Friendly Name                                        |
|   (30 octets)                                                 |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
                                ....
|                                                               |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

- **KNX medium**

  The KNX medium codes shall be as specified in Table 2.

Table 2 – KNX medium codes

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

These values shall be identical to the PID_MEDIUM_TYPE Property specified in [01]. the encoding is specified as DPT_Media in [03]; exactly one single bit shall be set.

♦ **Device Status**

Device Status octet shall be as specified in Figure 13.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	program ming mode

Figure 13 – Device Status

The encoding of bit 0 “programming mode” shall be identical to bit 0 “progmode” in PID_PROGMODE as specified in [01].

♦ **Project-Installation Identifier**

The Project-Installation identifier shall be as specified in Figure 14.

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

Figure 14 – Project-Installation Identifier

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 15 Lines, or in an environment with more than one KNX project.

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

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

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

The KNXnet/IP device routing multicast address shall be the multicast address that shall be 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. This information may be used if KNXnet/IP Routing frames 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 shall be 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 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 shall be filled with the NULL (00h) character.

7.5.4.3 Supported service families DIB

The supported service families DIB shall have the structure as specified in Figure 15.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Service Family ID           |   Service Family version      |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Service Family ID           |   Service Family version      |
|   (1 Octet)                   |   (1 Octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   ....                        |   ....                        |
|                               |                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Service Family ID           |   Service Family version      |
|   (1 Octet)                   |   (1 Octet)                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 15 – Supported services families DIB

The service family IDs shall be the high octet of the Service Type ID. A list of service type IDs can be found in [04].

The version of a service family shall refer to the version of the corresponding KNXnet/IP document. This version is only updated when the document itself is updated after it has gone through the KNX Association’s approval 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 shall be an integer. It does not represent the manufacturer’s implementation version.

7.5.4.4 IP Config DIB

The IP configuration DIB shall have the following structure.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   IP Address                                                  |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Subnet Mask                                                 |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Default Gateway                                             |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   IP Capabilities             |   IP assignment method        |
|   (1 Octet)                   |   (1 Octet)                   |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
```

Figure 16 – IP Config DIB

7.5.4.5 IP Current Config DIB

The IP current configuration DIB shall have the following structure.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
 |   Current IP Address                                        |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Current Subnet Mask                                         |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Current Default Gateway                                     |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DHCP Server                                                 |
|   (4 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                                                               |
|                                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+- - - - - - - - - - - - - - - -+
|  Current IP assignment method |   Reserved                    |
|  (1 Octet)                    |   (1 Octet)                   |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
```

Figure 17 - IP Current Config DIB

7.5.4.6 KNX Addresses DIB

The KNX address DIB shall have the following structure.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNX Individual Address                                      |
|   (2 octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Additional Individual Address 1 (optional)                  |
|   (2 octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Additional Individual Address 2 (optional)                  |
|   (2 octets)                                                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|   ...                                                         |
|                                                               |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
```

Figure 18 - KNX Addresses DIB

7.5.4.7 Manufacturer data DIB

The manufacturer data DIB has this structure.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Description Type Code       |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   KNX Manufacturer ID                                         |
|   (2 Octets)                                                  |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Any manufacturer specific data                              |
|   (?? Octets)                                                 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 19 – 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 device KNX Serial Number (6 octets).

The manufacturer data DIB may contain any manufacturer specific data.

7.6 Discovery

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 connectionless and stateless, the KNXnet/IP Client's discovery endpoint address information shall be included in the KNXnet/IP body.

```
                         KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   SEARCH_REQUEST                                              |
|   (0201h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(HPAI)                               |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                          KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Discovery endpoint                                          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 20 – SEARCH_REQUEST frame binary format

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 KNXnet/IP Client's discovery endpoint using the HPAI included in the received SEARCH_REQUEST frame.

The HPAI of the KNXnet/IP 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 KNXnet/IP Server shall announce each of its own control endpoints in a single SEARCH_RESPONSE frame.

```
                     KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   SEARCH_RESPONSE                                             |
|   (0202h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(HPAI) + sizeof(Description)         |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                      KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Control endpoint                                            |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   DIB                                                         |
|   device hardware                                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   DIB                                                         |
|   supported service families                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 21 – SEARCH_RESPONSE frame binary format

7.7 Self description

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 KNXnet/IP Client’s control endpoint.

```
                     KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DESCRIPTION_REQUEST                                         |
|   (0203h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(HPAI)                               |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                      KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Control endpoint                                            |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 22 – DESCRIPTION_REQUEST frame binary format

7.7.2 DESCRIPTION_RESPONSE

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 KNXnet/IP Client’s control endpoint using the HPAI included in the received DESCRIPTION_REQUEST frame.

The size of the KNXnet/IP body varies depending on the number of DIB structures sent by the KNXnet/IP Server in response to the KNXnet/IP Client’s DESCRIPTION_REQUEST.

```
                        KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DESCRIPTION_RESPONSE                                        |
|   (0204h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(Description)                        |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                         KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DIB                                                         |
|   device hardware                                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   DIB                                                         |
|   supported service families                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   DIB                                                         |
|   other device information (optional)                         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 23 – DESCRIPTION_RESPONSE frame binary format

7.8 Connection management

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 KNXnet/IP 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 3 - Connection types

Connection type	Value	V.	Description
DEVICE_MGMT_CONNECTION	03h	1	Data connection used to configure a KNXnet/IP device.
TUNNEL_CONNECTION	04h	1	Data connection used to forward KNX telegrams between two KNXnet/IP devices.
REMLOG_CONNECTION	06h	1	Data connection used for configuration and data transfer with a remote logging server.
REMCONF_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 KNXnet/IP Client’s data endpoint meant for the requested data connection shall complete the body of the CONNECT_REQUEST frame.

```
                          KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   CONNECT_REQUEST                                             |
|   (0205h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
| HEADER_SIZE_10 + sizeof(HPAI) + sizeof(HPAI) + sizeof(CRI)    |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                           KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Control endpoint                                            |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Data endpoint                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   CRI                                                         |
|   Connection request Information                              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 24 – CONNECT_REQUEST frame binary format

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 KNXnet/IP Client's control endpoint using the HPAI included in the received CONNECT_REQUEST frame.

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

If the connection request is successfully fulfilled with all the requested options, the body of the CONNECT_REQUEST frame shall contain a communication channel ID that shall uniquely identify 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 4 - Common CONNECT_RESPONSE status information

Error constant	Value	V.	Description
E_NO_ERROR	00h	1	The connection is 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 cannot accept the new data connection because its maximum amount of concurrent connections is already occupied.

The Host Protocol Address Information of the KNXnet/IP 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.

```
                           KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   CONNECT_RESPONSE                                            |
|   (0206h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 2 + sizeof(HPAI) + sizeof(CRD)             |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                            KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Communication Channel ID    |   Status                      |
|                               |                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Data endpoint                                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   CRD                                                         |
|   Connection Response Data Block                              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 25 – CONNECT_RESPONSE frame binary format

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 KNXnet/IP Client's control endpoint shall be added after the communication channel ID.

```
                           KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   CONNECTIONSTATE_REQUEST                                     |
|   (0207h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 2 + sizeof(HPAI)                           |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                            KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Communication Channel ID    |   reserved                    |
|                               |                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Control endpoint                                            |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 26 – CONNECTIONSTATE_REQUEST frame binary format

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 KNXnet/IP Client's control endpoint using the HPAI included in the received CONNECTIONSTATE_REQUEST frame.

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

The second octet of the KNXnet/IP body shall contain the status information of the connection state request. Table 5 lists the status codes that are defined for the CONNECTIONSTATE_RESPONSE frame.

Table 5 - 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 cannot find an active data connection with the specified ID.
E_DATA_CONNECTION	26h	1	The KNXnet/IP Server device detects an error concerning the data connection with the specified ID.
E_KNX_CONNECTION	27h	1	The KNXnet/IP Server device detects an error concerning the KNX subnetwork connection with the specified ID.

```
                         KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   CONNECTIONSTATE_RESPONSE                                    |
|   (0208h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 2                                          |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                          KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Communication Channel ID    |   Status                      |
|                               |                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 27 – CONNECTIONSTATE_RESPONSE frame binary format

7.8.5 DISCONNECT_REQUEST

```
                         KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DISCONNECT_REQUEST                                          |
|   (0209h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 2 + sizeof(HPAI)                           |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                          KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Communication Channel ID    |   reserved                    |
|                               |                               |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HPAI                                                        |
|   Control endpoint                                            |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 28 – DISCONNECT_REQUEST frame binary format

7.8.6 DISCONNECT_RESPONSE

```
                         KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   DISCONNECT_RESPONSE                                         |
|   (020Ah)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 2                                          |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                          KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Communication Channel ID    |   Status                      |
|                               |                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 29 – DISCONNECT_RESPONSE frame binary format

8 IP Networks

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 specification 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.

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 telegrams 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 telegrams. 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 communication 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 must 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 frame is transported on an IP channel, an IP message encapsulating the KNX frame 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.

8.3 Transport mechanisms

IP is a network level protocol. It is designed to operate over a wide range of physical media and Data Link Layer protocols. As such this document does not specify anything about the Data Link Layer or Physical Layers of the IP stack.

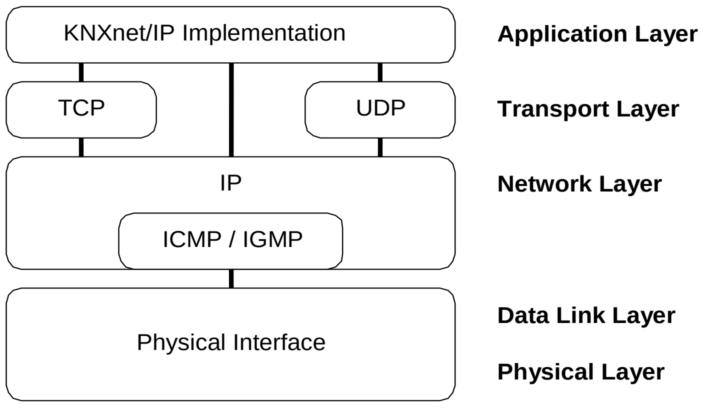

Figure 30 – IP protocol stack

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.

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 get 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 must be received reliably.

TCP has the advantage of reliable delivery service and hence will guarantee that the received packet ordering is preserved. On the down side it does not support multicast 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 frames, 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 frames and its support of multicast 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 them.

Using UDP, datagrams can be sent using either unicast or multicast 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 multicast addressing. Therefore it is recommended that KNXnet/IP devices support both unicast and multicast IP addressing although it is not required that a KNXnet/IP device supports multicasts in order to interoperate with a KNXnet/IP device that does.

8.5 IP Address Assignment

8.5.1 IP unicast address

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.

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.

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.

8.5.1.4 IP address assignment procedure

Figure 31 describes the address assignment procedure for KNXnet/IP devices.

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 to.
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 to.
5	IF	AutoIP address assignment is successful	THEN	use newly assigned IP address.
			ELSE	use fixed IP address already assigned to.
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.

Figure 31 – Address assignment procedure

8.5.2 IP multicast addresses

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.

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 shall have 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.

8.6 KNXnet/IP host protocol

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 multicast group, and may be capable of belonging to up to 16 multicast groups. A KNXnet/IP device shall support multicasting.

A KNXnet/IP device shall support these IP protocols: ARP, ICMP, IGMP, BootP / DHCP [4)] 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 a fixed port number shall be defined 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.

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, shall be stored binary in network octet order.

```
+-------------------------------+-------------------------------+
|               8               |   Host Protocol Code          |
|                               |   (1 octet)                   |
+-------------------------------+-------------------------------+
|                          IP Address                           |
|                          (4 octets)                           |
+---------------------------------------------------------------+
|                        IP port number                         |
|                          (2 octets)                           |
+---------------------------------------------------------------+
```

Figure 32 – IP Host Protocol Address Information binary format

[4] BootP / DHCP: Either one shall be implemented.

Table 6 – 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.

8.6.3 KNXnet/IP Endpoints

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.

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

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.

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 frame. A KNXnet/IP server 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 KNXnet/IP Client does not change this port either.

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.

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.

8.7 General implementation guidelines

Refer to [04] for general IP implementation guidelines.

8.8 Binary examples of KNXnet/IP IP frames

8.8.1 SEARCH_REQUEST

```
         +-------------------------------+
  1      |             06h               |    header size
         +-------------------------------+
  2      |             10h               |    protocol version
         +-------------------------------+
  3      |             02h               | \
         +- - - - - - - - - - - - - - - -+  > service type identifier 0201h
  4      |             01h               | /
         +-------------------------------+
  5      |             00h               | \
         +- - - - - - - - - - - - - - - -+  > total length, 14 octets
  6      |             0Eh               | /
         +-------------------------------+
  7      |             08h               |    structure length
         +-------------------------------+
  8      |             01h               |    host protocol code, e.g. 01h, for UDP over IPv4
         +-------------------------------+
  9      |             192               | \
         +- - - - - - - - - - - - - - - -+ |
 10      |             168               | |
         +- - - - - - - - - - - - - - - -+  > IP address of control endpoint,
 11      |             200               | |   e.g. 192.168.200.12
         +- - - - - - - - - - - - - - - -+ |
 12      |              12               | /
         +-------------------------------+
 13      |             0Eh               | \
         +- - - - - - - - - - - - - - - -+  > port number of control endpoint, 3671
 14      |             57h               | /
         +-------------------------------+
```

Figure 33 – SEARCH_REQUEST frame binary format: IP example

8.8.2 SEARCH_RESPONSE

```
         +-------------------------------+
   1     |              06h              |    header size
         +-------------------------------+
   2     |              10h              |    protocol version
         +-------------------------------+
   3     |              02h              | \
         +- - - - - - - - - - - - - - - -+  > service type identifier 0202h
   4     |              02h              | /
         +-------------------------------+
   5     |              00h              | \
         +- - - - - - - - - - - - - - - -+  > total length, 78 octets
   6     |              4Eh              | /
         +-------------------------------+
   7     |              08h              |    structure length (HPAI)
         +-------------------------------+
   8     |              01h              |    host protocol code, e.g. 01h, for UDP over IPv4
         +-------------------------------+
   9     |              192              | \
         +- - - - - - - - - - - - - - - -+ |
  10     |              168              | |
         +- - - - - - - - - - - - - - - -+  > IP address of control endpoint,
  11     |              200              | |   e.g. 192.168.200.12
         +- - - - - - - - - - - - - - - -+ |
  12     |               12              | /
         +-------------------------------+
  13     |              C3h              | \
         +- - - - - - - - - - - - - - - -+  > port number of control endpoint,
  14     |              B4h              | /   e.g. 50100
         +-------------------------------+
  15     |              36h              |    structure length (DIB hardware)
         +-------------------------------+
  16     |              01h              |    description type code, 01h = DEVICE_INFO
         +-------------------------------+
  17     |              02h              |    KNX medium, 02h = TP1 (KNX TP)
         +-------------------------------+
  18     |              01h              |    Device Status, 01h = programming mode
         +-------------------------------+
  19     |              11h              | \
         +- - - - - - - - - - - - - - - -+  > KNX Individual Address, e.g. 1.1.0
  20     |              00h              | /
         +-------------------------------+
  21     |              00h              | \
         +- - - - - - - - - - - - - - - -+  > Project-Installation ID, e.g. 0011h
  22     |              11h              | /
         +-------------------------------+
  23     |              00h              | \
         +- - - - - - - - - - - - - - - -+ |
  24     |              01h              | |
         +- - - - - - - - - - - - - - - -+ |
  25     |              11h              | |
         +- - - - - - - - - - - - - - - -+  > KNX device KNX Serial Number,
  26     |              11h              | |
         +- - - - - - - - - - - - - - - -+ |
  27     |              11h              | |
         +- - - - - - - - - - - - - - - -+ |
  28     |              11h              | /
         +-------------------------------+
  29     |              224              | \
         +- - - - - - - - - - - - - - - -+ |
  30     |                0              | |
         +- - - - - - - - - - - - - - - -+  > device routing multicast address
  31     |               23              | |   e.g. 224.0.23.12
         +- - - - - - - - - - - - - - - -+ |
  32     |               12              | /
```

```
        +-------------------------------+
33      |              45h              | \
        +- - - - - - - - - - - - - - - -+ |
34      |              49h              | |
        +- - - - - - - - - - - - - - - -+ |
35      |              42h              | |
        +- - - - - - - - - - - - - - - -+  > KNXnet/IP MAC address
36      |              6Eh              | |
        +- - - - - - - - - - - - - - - -+ |
37      |              65h              | |
        +- - - - - - - - - - - - - - - -+ |
38      |              74h              | /
        +-------------------------------+
39      |              'M'              | \
        +- - - - - - - - - - - - - - - -+ |
40      |              'Y'              | |
        +- - - - - - - - - - - - - - - -+ |
41      |              'H'              | |
        +- - - - - - - - - - - - - - - -+  > Device Friendly Name, e.g. "MYHOME"
42      |              'O'              | |
        +- - - - - - - - - - - - - - - -+ |
43      |              'M'              | |
        +- - - - - - - - - - - - - - - -+ |
44      |              'E'              | |
        +- - - - - - - - - - - - - - - -+ |
45      |              00h              | |
        +- - - - - - - - - - - - - - - -+ |
..      |              ...              | |
        +- - - - - - - - - - - - - - - -+ |
68      |              00h              | /
        +-------------------------------+
69      |              0Ah              |    structure length (DIB supported service family)
        +-------------------------------+
70      |              02h              |    description type code, 02h = SUPP_SVC_FAMILIES
        +-------------------------------+
71      |              02h              |    service family, e.g. 02h = KNXnet/IP Core
        +-------------------------------+
72      |              01h              |    service family version, e.g. 01h
        +-------------------------------+
73      |              03h              |    service family, e.g. 03h = KNXnet/Device Mgmt
        +-------------------------------+
74      |              01h              |    service family version, e.g. 01h
        +-------------------------------+
75      |              04h              |    service family, e.g. 04h = KNXnet/IP Tunnelling
        +-------------------------------+
76      |              01h              |    service family version, e.g. 01h
        +-------------------------------+
77      |              05h              |    service family, e.g. 05h = KNXnet/IP Routing
        +-------------------------------+
78      |              01h              |    service family version, e.g. 01h
        +-------------------------------+
```

Figure 34 – SEARCH_RESPONSE frame binary format: IP example

8.8.3 DESCRIPTION_REQUEST

```
        +-------------------------------+
  1     |              06h              |    header size
        +-------------------------------+
  2     |              10h              |    protocol version
        +-------------------------------+
  3     |              02h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0203h
  4     |              03h              | /
        +-------------------------------+
  5     |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, 14 octets
  6     |              0Eh              | /
        +-------------------------------+
  7     |              08h              |    structure length
        +-------------------------------+
  8     |              01h              |    host protocol code, e.g. 01h, for UDP over IPv4
        +-------------------------------+
  9     |              192              | \
        +- - - - - - - - - - - - - - - -+ |
 10     |              168              | |
        +- - - - - - - - - - - - - - - -+  > IP address of control endpoint,
 11     |              200              | |   e.g. 192.168.200.12
        +- - - - - - - - - - - - - - - -+ |
 12     |               12              | /
        +-------------------------------+
 13     |              C3h              | \
        +- - - - - - - - - - - - - - - -+  > port number of control endpoint,
 14     |              B4h              | /   e.g. 50100
        +-------------------------------+
```

Figure 35 – KNXnet/IP DESCRIPTION_REQUEST frame binary format example

8.8.4 DESCRIPTION_RESPONSE

```
        +-------------------------------+
  1     |              06h              |    header size
        +-------------------------------+
  2     |              10h              |    protocol version
        +-------------------------------+
  3     |              02h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0204h
  4     |              04h              | /
        +-------------------------------+
  5     |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, 78 octets
  6     |              4Eh              | /
        +-------------------------------+
 7      |              36h              |    structure length (DIB hardware)
        +-------------------------------+
 8      |              01h              |    description type code, 01h = DEVICE_INFO
        +-------------------------------+
 9      |              02h              |    KNX medium, 02h = TP1 (KNX TP)
        +-------------------------------+
 10     |              01h              |    Device Status, 01h = programming mode
        +-------------------------------+
 11     |              11h              | \
        +- - - - - - - - - - - - - - - -+  > KNX Individual Address, e.g. 1.1.0
 12     |              00h              | /
        +-------------------------------+
 14     |              00h              | \
        +- - - - - - - - - - - - - - - -+  > Project-Installation ID, e.g. 0011h
 15     |              11h              | /
        +-------------------------------+
 16     |              00h              | \
        +- - - - - - - - - - - - - - - -+ |
 17     |              01h              | |
        +- - - - - - - - - - - - - - - -+ |
 18     |              11h              | |
        +- - - - - - - - - - - - - - - -+  > KNX device KNX Serial Number,
 19     |              11h              | |          includes manufacturer ID (2 octets)
        +- - - - - - - - - - - - - - - -+ |
 20     |              11h              | |
        +- - - - - - - - - - - - - - - -+ |
 21     |              11h              | /
```

```
          +-------------------------------+
  22      |             224               | \
          +- - - - - - - - - - - - - - - -+ |
  23      |               0               | |
          +- - - - - - - - - - - - - - - -+  > device routing multicast address
  24      |              23               | |   e.g. 224.0.23.12
          +- - - - - - - - - - - - - - - -+ |
  25      |              12               | /
          +-------------------------------+
  26      |             45h               | \
          +- - - - - - - - - - - - - - - -+ |
  27      |             49h               | |
          +- - - - - - - - - - - - - - - -+ |
  28      |             42h               | |
          +- - - - - - - - - - - - - - - -+  > KNXnet/IP MAC address
  29      |             6Eh               | |
          +- - - - - - - - - - - - - - - -+ |
  30      |             65h               | |
          +- - - - - - - - - - - - - - - -+ |
  31      |             74h               | /
          +-------------------------------+
  32      |             ‘M’               | \
          +- - - - - - - - - - - - - - - -+ |
  33      |             ‘Y’               | |
          +- - - - - - - - - - - - - - - -+ |
  34      |             ‘H’               | |
          +- - - - - - - - - - - - - - - -+  > Device Friendly Name, e.g. “MYHOME”
  35      |             ‘O’               | |
          +- - - - - - - - - - - - - - - -+ |
  36      |             ‘M’               | |
          +- - - - - - - - - - - - - - - -+ |
  37      |             ‘E’               | |
          +- - - - - - - - - - - - - - - -+ |
  38      |             00h               | |
          +- - - - - - - - - - - - - - - -+ |
  ..      |             ...               | |
          +- - - - - - - - - - - - - - - -+ |
  60      |             00h               | /
          +-------------------------------+
  61      |             0Ah               |    structure length (DIB supported service family)
          +-------------------------------+
  62      |             02h               |    description type code, 02h = SUPP_SVC_FAMILIES
          +-------------------------------+
  63      |             02h               |    service family, e.g. 02h = KNXnet/IP Core
          +-------------------------------+
  64      |             01h               |    service family version, e.g. 01h
          +-------------------------------+
  65      |             03h               |    service family, e.g. 03h = KNXnet/Device Mgmt
          +-------------------------------+
  66      |             01h               |    service family version, e.g. 01h
          +-------------------------------+
  67      |             04h               |    service family, e.g. 04h = KNXnet/IP Tunnelling
          +-------------------------------+
  68      |             01h               |    service family version, e.g. 01h
          +-------------------------------+
  69      |             05h               |    service family, e.g. 05h = KNXnet/IP Routing
          +-------------------------------+
  70      |             01h               |    service family version, e.g. 01h
          +-------------------------------+
  71      |             08h               |    structure length (DIB other device information)
          +-------------------------------+
  72      |             FEh               |    description type code, FEh = MFR_DATA
          +-------------------------------+
  73      |             00h               | \
          +- - - - - - - - - - - - - - - -+  > KNX Manufacturer ID, e.g. 0001h
  74      |             01h               | /
          +-------------------------------+
  75      |             ‘N’               | \
          +- - - - - - - - - - - - - - - -+ |
  76      |             ‘1’               | |
          +- - - - - - - - - - - - - - - -+  > Device Type Name, e.g. “N146”
  77      |             ‘4’               | |
          +- - - - - - - - - - - - - - - -+ |
  78      |             ‘6’               | /
          +-------------------------------+
```

Figure 36 – DESCRIPTION_RESPONSE frame binary format: IP example

8.8.5 CONNECT_REQUEST

```
          +-------------------------------+
   1      |              06h              |    header size
          +-------------------------------+
   2      |              10h              |    protocol version
          +-------------------------------+
   3      |              02h              | \
          +- - - - - - - - - - - - - - - -+  > service type identifier 0205h
   4      |              05h              | /
          +-------------------------------+
   5      |              00h              | \
          +- - - - - - - - - - - - - - - -+  > total length, 24 octets
   6      |              1Ah              | /
          +-------------------------------+
   7      |              08h              |    structure length
          +-------------------------------+
   8      |              01h              |    host protocol code, e.g. 01h, for UDP over IPv4
          +-------------------------------+
   9      |              192              | \
          +- - - - - - - - - - - - - - - -+ |
  10      |              168              | |
          +- - - - - - - - - - - - - - - -+  > IP address of control endpoint,
  11      |              200              | |    e.g. 192.168.200.12
          +- - - - - - - - - - - - - - - -+ |
  12      |               12              | /
          +-------------------------------+
  13      |              C3h              | \
          +- - - - - - - - - - - - - - - -+  > port number of control endpoint,
  14      |              B4h              | /    e.g. 50100
          +-------------------------------+
  15      |              08h              |    structure length
          +-------------------------------+
  16      |              01h              |    host protocol code, e.g. 01h, for UDP over IPv4
          +-------------------------------+
  17      |              192              | \
          +- - - - - - - - - - - - - - - -+ |
  18      |              168              | |
          +- - - - - - - - - - - - - - - -+  > IP address of data endpoint,
  19      |              200              | |    e.g. 192.168.200.20
          +- - - - - - - - - - - - - - - -+ |
  20      |               20              | /
          +-------------------------------+
  21      |              C3h              | \
          +- - - - - - - - - - - - - - - -+  > port number of data endpoint,
  22      |              B4h              | /    e.g. 50100
          +-------------------------------+
  23      |              04h              |    structure length
          +-------------------------------+
  24      |              04h              |    connection type code, e.g. 04h, TUNNEL_CONNECTION
          +-------------------------------+
  25      |              02h              |    KNX layer, e.g. TUNNEL_LINKLAYER
          +-------------------------------+
  26      |              00h              |    reserved
          +-------------------------------+
```

Figure 37 – KNXnet/IP CONNECT_REQUEST frame binary format example

8.8.6 CONNECT_RESPONSE

```
        +-------------------------------+
  1     |              06h              |    header size
        +-------------------------------+
  2     |              10h              |    protocol version
        +-------------------------------+
  3     |              02h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0206h
  4     |              06h              | /
        +-------------------------------+
  5     |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, 20 octets
  6     |              14h              | /
        +-------------------------------+
  7     |              15h              |    communication channel ID, e.g. 21
        +-------------------------------+
  8     |              00h              |    status code (NO_ERROR)
        +-------------------------------+
  9     |              08h              |    structure length
        +-------------------------------+
 10     |              01h              |    host protocol code, e.g. 01h, for UDP over IPv4
        +-------------------------------+
 11     |              192              | \
        +- - - - - - - - - - - - - - - -+ |
 12     |              168              | |
        +- - - - - - - - - - - - - - - -+  > IP address of data endpoint,
 13     |              200              | |    e.g. 192.168.200.20
        +- - - - - - - - - - - - - - - -+ |
 14     |               20              | /
        +-------------------------------+
 15     |              C3h              | \
        +- - - - - - - - - - - - - - - -+  > port number of data endpoint,
 16     |              B4h              | /    e.g. 50100
        +-------------------------------+
 17     |              04h              |   structure length of CRD for TUNNELING_CONNECTION
        +-------------------------------+
 18     |              04h              |  connection type code, e.g. 04h, TUNNEL_CONNECTION
        +-------------------------------+
 19     |              11h              | \
        +- - - - - - - - - - - - - - - -+  > Individual Address, e.g. 01.01.10,
 20     |              0Ah              | /  used for TUNNELING_CONNECTION
        +-------------------------------+
```

Figure 38 – CONNECT_RESPONSE frame binary format: IP example

8.8.7 CONNECTIONSTATE_REQUEST

```
       +-------------------------------+
  1    |              06h              |    header size
       +-------------------------------+
  2    |              10h              |    protocol version
       +-------------------------------+
  3    |              02h              | \
       +- - - - - - - - - - - - - - - -+  > service type identifier 0207h
  4    |              07h              | /
       +-------------------------------+
  5    |              00h              | \
       +- - - - - - - - - - - - - - - -+  > total length, 16 octets
  6    |              10h              | /
       +-------------------------------+
  7    |              15h              |    communication channel ID, e.g. 21
       +-------------------------------+
  8    |              00h              |    reserved
       +-------------------------------+
  9    |              08h              |    structure length
       +-------------------------------+
 10    |              01h              |    host protocol code, e.g. 01h, for UDP over IPv4
       +-------------------------------+
 11    |              192              | \
       +- - - - - - - - - - - - - - - -+ |
 12    |              168              | |
       +- - - - - - - - - - - - - - - -+  > IP address of control endpoint,
 13    |              200              | |   e.g. 192.168.200.12
       +- - - - - - - - - - - - - - - -+ |
 14    |               12              | /
       +-------------------------------+
 15    |              C3h              | \
       +- - - - - - - - - - - - - - - -+  > port number of control endpoint,
 16    |              B4h              | /   e.g. 50100
       +-------------------------------+
```

Figure 39 – CONNECTIONSTATE_REQUEST frame binary format: IP example

8.8.8 CONNECTIONSTATE_RESPONSE

```
       +-------------------------------+
  1    |              06h              |    header size
       +-------------------------------+
  2    |              10h              |    protocol version
       +-------------------------------+
  3    |              02h              | \
       +- - - - - - - - - - - - - - - -+  > service type identifier 0208h
  4    |              08h              | /
       +-------------------------------+
  5    |              00h              | \
       +- - - - - - - - - - - - - - - -+  > total length, 8 octets
  6    |              08h              | /
       +-------------------------------+
  7    |              15h              |    communication channel ID, e.g. 21
       +-------------------------------+
  8    |              00h              |    status code (NO_ERROR)
       +-------------------------------+
```

Figure 40 – CONNECTIONSTATE_RESPONSE frame binary format: IP example

8.8.9 DISCONNECT_REQUEST

```
          +-------------------------------+
    1     |             06h               |    header size
          +-------------------------------+
    2     |             10h               |    protocol version
          +-------------------------------+
    3     |             02h               | \
          +- - - - - - - - - - - - - - - -+  > service type identifier 0209h
    4     |             09h               | /
          +-------------------------------+
    5     |             00h               | \
          +- - - - - - - - - - - - - - - -+  > total length, 16 octets
    6     |             10h               | /
          +-------------------------------+
    7     |             15h               |    communication channel ID, e.g. 21
          +-------------------------------+
    8     |             00h               |    reserved
          +-------------------------------+
    9     |             08h               |    structure length
          +-------------------------------+
   10     |             01h               |    host protocol code, e.g. 01h, for UDP over IPv4
          +-------------------------------+
   11     |             192               | \
          +- - - - - - - - - - - - - - - -+ |
   12     |             168               | |
          +- - - - - - - - - - - - - - - -+  > IP address of control endpoint,
   13     |             200               | |   e.g. 192.168.200.12
          +- - - - - - - - - - - - - - - -+ |
   14     |              12               | /
          +-------------------------------+
   15     |             C3h               | \
          +- - - - - - - - - - - - - - - -+  > port number of control endpoint,
   16     |             B4h               | /   e.g. 50100
          +-------------------------------+
```

Figure 41 – DISCONNECT_REQUEST frame binary format: IP example

8.8.10 DISCONNECT_RESPONSE

```
          +-------------------------------+
    1     |             06h               |    header size
          +-------------------------------+
    2     |             10h               |    protocol version
          +-------------------------------+
    3     |             02h               | \
          +- - - - - - - - - - - - - - - -+  > service type identifier 020Ah
    4     |             0Ah               | /
          +-------------------------------+
    5     |             00h               | \
          +- - - - - - - - - - - - - - - -+  > total length, 8 octets
    6     |             08h               | /
          +-------------------------------+
    7     |             15h               |    communication channel ID, e.g. 21
          +-------------------------------+
    8     |             00h               |    status code (NO_ERROR)
          +-------------------------------+
```

Figure 42 – DISCONNECT_RESPONSE frame binary format: IP example

9 Certification

9.1 Introduction

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

9.2 Supported services

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

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

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

Service Name	Client	Server
IPV4_UDP	R	R
IPV4_TCP	O	O
IPV6_UDP	to de defined	to be defined
IPV6_TCP	to be defined	to be defined

Legend: “R” = Required, “O” = Optional, “n.a.” = not applicable