System Specifications

KNXnet/IP

Tunnelling

3

8

4

Summary

This document provides the KNXnet/IP Tunnelling specification.

Version 01.05.03 is a KNX Approved Standard.

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

Document updates

Version	Date	Modifications
1.0 DP	2004.01.07	Final version for Release for Voting
1.1 DV	2005.05.30	Draft for Voting for Final Voting.
1.3 AS	2008.07.02	Publication of the Approved Standard.
1.4 AS	2008.09.05	• **AN110 "Phasing out A-Mode"** integrated
1.4 AS	2009.06.29	Editorial update in view of inclusion in the KNX Specifications v2.0.
01.05.00	2013.07.18	• **AN139 "Procedures for the assignment of IAs to KNXnet/IP Tunnelling connections"** integrated.
01.05.01	2013.07.18	• Editorial review.
01.05.01	2013.09.20	• 3.2.3.5.3 "Assignment via PID_INDIVIDUAL_ADDRESS": PID_KNX_INDIVIDUAL_ADDRESS is located in the KNXnet/IP Parameter Object, and not in the Device Object.
01.05.03	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.

[1] Chapter 3/5/2 "Management Procedures"
[2] Chapter 3/5/3 "Configuration Procedures"
[3] Chapter 3/6/3 "External Message Interface"
[4] Chapter 3/8/1 "Overview" (KNXnet/IP)
[5] Chapter 3/8/2 "Core" (KNXnet/IP)
[6] Chapter 3/8/3 "Device Management" (KNXnet/IP)
[7] Chapter 3/8/3 "Remote Configuration and Diagnosis" (planned)

Filename:	03_08_04 Tunnelling v01.05.03 AS.docx
Version:	01.05.03
Status:	Approved Standard
Savedate:	2013.10.28
Number of pages:	20

1) Request for Comment: Internet Standards defined by the Internet Engineering Task Force (IETF) are firstly 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 [4].

General frame descriptions and data exchange protocols between KNXnet/IP devices are described in [5].

General device management and configuration of KNXnet/IP devices is specified in [6].

This Chapter 3/8/4 “Tunnelling” of the KNXnet/IP specification describes point-to-point exchange of KNX telegrams over an IP network between an KNXnet/IP device acting as a server and an KNXnet/IP Client for configuration and diagnostics. KNX frames are encapsulated inside IP datagrams. KNXnet/IP Tunnelling does not address timing issues caused by IP data network latency greater than one second. Refer to [7] (planned).

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

1.2 Definitions, acronyms and abbreviations

Refer to [4] for a list of definitions for the KNXnet/IP specification.

2 Tunnelling of KNX frames

2.1 Introduction

Tunnelling is characterized by ETS sending a single KNX frame in an IP frame and waiting until the response arrives or a time-out is reached.

2.2 Tunnelling

2.2.1 General requirements

KNX frames shall always be sent within a TUNNELLING_REQUEST frame. This frame shall contain the KNX frame in cEMI format. The cEMI format shall be supported by all KNXnet/IP devices.

After a communication channel has been established, the following KNX services shall be supported by a KNXnet/IP version 1 implementation:

KNXnet/IP Tunnelling on KNX Data Link Layer

Client → Server	L_Data.req, M_Reset.req
Server → Client	L_Data.con, L_Data.ind

KNXnet/IP Tunnelling in cEMI Raw mode

Client → Server	L_Raw.req, M_Reset.req
Server → Client	L_Raw.con, L_Raw.ind

KNXnet/IP Tunnelling on KNX Busmonitor

Client → Server	n.a.
Server → Client	L_Busmon.ind

2.2.2 Tunnelling on KNX Data Link Layer

Implementation of Tunnelling on KNX Data Link Layer is mandatory.

Each KNXnet/IP Tunnelling connection shall correspond with a KNX Individual Address i.e. when the Tunnelling connection is established the KNXnet/IP Server shall assign a KNX Individual Address to it.

This Tunnelling Individual Address shall be unique within the open KNXnet/IP Tunnelling connections of the KNXnet/IP Tunnelling device. The KNXnet/IP Tunnelling Server shall not open two or more KNXnet/IP Tunnelling connections with the same Tunnelling Individual Address.

In case a KNXnet/IP Tunnelling client connection request cannot be accepted because the list of unique Individual Addresses has been depleted and all remaining Additional Individual Address entries are identical and one of these entries is already in use, then the KNXnet/IP Tunnelling Server SHALL return the error code NO_MORE_¬UNIQUE_-CONNECTIONS (= 25h) as the CONNECT_RESPONSE status code. This error code is mandatory for any device that offers more than one KNXnet/IP Tunnelling connection.

The KNXnet/IP Tunnelling Client side error handling is specified in 3.3.

This KNX Individual Address shall be returned in the CONNECT_RESPONSE frame Connection Response Data Block (CRD). If the KNXnet/IP Server assigns its own KNX Individual Address to the Tunnelling connection then management of the KNXnet/IP Server shall not be possible via KNXnet/IP Tunnelling frames or from the KNX Subnetwork. The KNX Individual Address of the KNXnet/IP Server itself shall be obtained by assignment through ETS. This address may be used for a KNXnet/IP Tunnelling connection with the implication stated above [see Figure 1-A].

KNX Individual Addresses for any additional Tunnelling connection shall be assigned by ETS.

The additional KNX Individual Addresses shall be stored in Property PID_ADDITIONAL_-INDIVIDUAL_ADDRESSES.

To prevent other Management Clients from using or assigning any of these additional IAs the KNXnet/IP Tunnelling device shall defend its additional IA. Such Management Client will to this execute the Management Procedure NM_IndividualAddress_Check (see [1]). If this Management Client in this request tries to establish a Transport Layer connection to any of these IAs by sending a T_Connect-PDU, then the KNXnet/IP Tunnelling device shall act as follows.

- If <u>no Tunnelling connection</u> is open for the additional KNX Individual Address, then the Tunnelling device shall send a T_Disconnect-PDU.
- If <u>the Tunnelling connection for this additional IA is currently open</u>, then the Tunnelling device shall forward the above request as T_Connect-PDU to the connected KNXnet/IP Tunnelling Client. The KNXnet/IP Tunnelling Client then shall respond to the request.

Additional KNX Individual Addresses shall be permanent. The KNXnet/IP Server shall generate Layer-2 acknowledge frames for these additionally assumed KNX Individual Addresses [see Figure 1-B] 2).

If a KNXnet/IP Server also implements KNXnet/IP Routing it shall not use its own KNX Individual Address for KNXnet/IP Tunnelling connections but shall assume KNX Individual Addresses as described above [see Figure 1-C].

A KNXnet/IP Router shall not activate KNXnet/IP Tunnelling until at least one additional KNX Individual Address has been set via KNXnet/IP Device Management.

These are the bootstrap steps to take for a KNXnet/IP Router.

a) KNXnet/IP Discovery of KNXnet/IP Router IP address

b) Initiation of KNXnet/IP Device Management connection to KNXnet/IP Router and setting KNX Individual Address for this KNXnet/IP Router.

c) Setting KNXnet/IP Router additional KNX Individual Address(es)

The KNXnet/IP Tunnelling Server shall only pass those KNX point-to-point addressed telegrams to the KNXnet/IP Tunnelling Client that contain the KNX Individual Address of the KNXnet/IP Tunnelling connection with this KNXnet/IP Tunnelling Client. All KNX telegrams on KNX point-to-multipoint communication (i.e. group addressing) shall be forwarded to the KNXnet/IP Tunnelling Client.

If the KNXnet/IP Tunnelling Client sends a cEMI frame L_Data.req with a KNX Individual Address (KNX Source Address) set to 0000h then the Tunnelling server shall enter the KNX Individual Address assigned to this Tunnelling connection. Otherwise the KNX frame shall be sent unchanged (see cEMI specifications in [3]).

2) Every KNXnet/IP Tunnelling connection shall use its own (unique) Individual Address and hence logically appears as another device on the KNX bus. Hence, it shall acknowledge telegrams even if the telegram is coming from the physically same device.

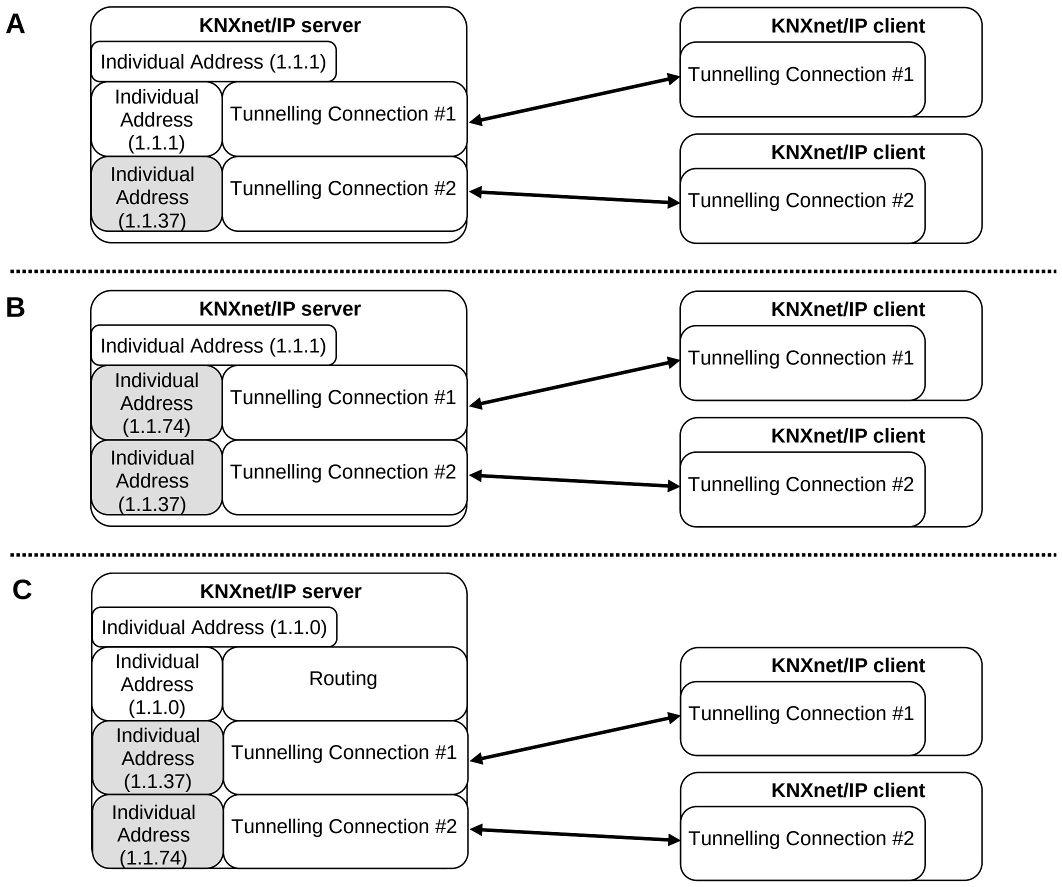

Figure 1 – Tunnelling connections and KNX Individual Addresses in the KNXnet/IP Server

2.2.3 Tunnelling in cEMI Raw mode

Implementation of Tunnelling on KNX cEMI Raw mode is optional.

The KNXnet/IP Tunnelling Server shall pass any KNX telegram received to the KNXnet/IP Tunnelling Client.

The KNXnet/IP Tunnelling Server shall not generate Layer-2 acknowledge frames in response to KNX telegrams forwarded onto a KNXnet/IP Tunnelling connection in cEMI Raw mode.

Because KNXnet/IP Tunnelling on cEMI Raw mode would therefore disable the KNXnet/IP Routing function in a KNXnet/IP Router, KNXnet/IP Tunnelling on cEMI Raw mode shall not be supported in a KNXnet/IP Routing device.

2.2.4 Tunnelling on KNX Busmonitor

Implementation of Tunnelling on KNX Busmonitor is optional.

If Tunnelling on KNX Busmonitor is implemented the KNXnet/IP Tunnelling Server shall only support one KNXnet/IP Tunnelling connection per KNX Subnetwork. If Tunnelling on KNX Busmonitor is activated then the KNXnet/IP Tunnelling Server may not support any other KNXnet/IP services for the KNX Subnetwork.

Because KNXnet/IP Tunnelling on KNX Busmonitor would therefore disable the KNXnet/IP Routing function in a KNXnet/IP Router, KNXnet/IP Tunnelling on KNX Busmonitor shall not be supported in a KNXnet/IP Routing device.

2.3 Timing

KNXnet/IP Tunnelling as such does *not* provide any mechanisms to modify timing requirements on the tunnelled data packets. More precisely, the timing requirements defined by the KNX specification are still valid while tunnelling or routing over a non-KNX network. Therefore the transfer time of tunnelled data packets shall allow operation within these timing requirements.

This implies that $t_{Tunnelling\_protocol\_transfer\_time} << t_{KNX\_transfer\_time-outs}$.

2.4 Sending KNX frames via KNXnet/IP Tunnelling Server to local subnet

To send a KNX frame, the KNXnet/IP Tunnelling Client shall send a TUNNELLING_REQUEST frame to the KNXnet/IP Tunnelling Server, with an L_Data.req, according to the KNX specification. The KNXnet/IP Tunnelling Client shall be connected and shall not be in Busmonitor Mode.

2.5 Receiving KNX frames via KNXnet/IP Tunnelling Server from local subnet

If a connection is established, the KNXnet/IP Tunnelling Server shall send a TUNNELLING_REQUEST frame for each telegram it receives from the local KNX Subnetwork. The KNX services of the embedded KNX frame may be L_Data.con, L_Data.ind, or L_Busmon.ind, according to the KNX specification.

2.6 Frame confirmation

TUNNELLING_REQUEST frames shall be confirmed using TUNNELLING_ACK frames.

If a TUNNELLING_REQUEST frame is not confirmed within the TUNNELLING_REQUEST_TIME_-OUT time of one (1) second then the frame shall be repeated once with the same sequence counter value by the sending KNXnet/IP device.

If the KNXnet/IP device does not receive a TUNNELLING_ACK frame within the TUNNELLING_-REQUEST_TIMEOUT (= 1 second) or the status of a received TUNNELLING_ACK frame signals any kind of error condition, the sending device shall repeat the TUNNELLING_REQUEST frame once and then terminate the connection by sending a DISCONNECT_REQUEST frame to the other device's control endpoint.

If a KNXnet/IP Tunnelling Server receives a data packet with a sequence number that is the expected sequence number then it shall reply with a TUNNELLING_ACK (Status = E_NO_ERROR) frame and process the received frame.

If a KNXnet/IP Tunnelling Server receives a frame with a sequence number that is one less than the expected sequence number then it shall reply with a TUNNELLING_ACK (Status = E_NO_ERROR) frame and discard the received frame.

If a KNXnet/IP Tunnelling Server receives a data packet with a sequence number that is not equal to the expected sequence number and not equal to one less than the expected sequence number, then the KNXnet/IP Tunnelling Server shall not reply and shall discard the received frame.

If a KNXnet/IP Tunnelling Client receives a frame with a sequence number that equals the expected sequence number then it shall reply with a TUNNELLING_ACK (Status = E_NO_ERROR) frame and process the received frame.

If a KNXnet/IP Tunnelling Client receives a frame with a sequence number that is one less than the expected sequence number then it shall reply with a TUNNELLING_ACK (Status = E_NO_ERROR) message and discard the received frame.

If a KNXnet/IP Tunnelling Client receives a frame with a sequence number that is not equal to the expected sequence number and not equal to one less than the expected sequence number, the KNXnet/IP Tunnelling Client shall not reply and shall discard the received frame.

3 Configuration and Management

3.1 General

General device management and configuration of KNXnet/IP devices is specified in [6].

KNXnet/IP Tunnelling does not require any configuration beyond the general device management.

3.2 IA assignment for KNXnet/IP Tunnelling connections

3.2.1 Goal

The goal of this procedure is to change the Individual Address used for a KNXnet/IP Tunnelling Connection to an Individual Address IA_{req} provided by the ETS user.

3.2.2 Prerequisites

1. The KNXnet/IP Tunnelling Client shall not assign an IA_{req} for which the Device Address part is 00h.

   NOTE 1 The responsibility for this lies with the Management Client. In the Management Server definition for the Individual Address and PID_ADDITIONAL_INDIVIDUAL_ADDRESSES no error handling is foreseen for values of any entry with Device Address 00h.

2. Form 2.2.2: a KNXnet/IP Router shall not use its own IA as KNXnet/IP Tunnelling IA. However, other IAs with Device Address part 00h can still be used if these are not used by other Couplers in the installation. The Device Address part of CRD.IA can thus be 00h.
   (CRD.IA is at any time the IA assigned to the KNXnet/IP Tunnelling Connection.)

   NOTE 2 This second prerequisite is mainly intended to take into account KNXnet/IP Tunnelling Clients that do not respect the first prerequisite. It is not a wanted situation that an additional Tunnelling Address in a KNXnet/IP Router has the Device Address part 00h. The Tunnelling Address would then be topologically not be correct, because the KNXnet/IP Tunnelling Server is logically linked to the Subnetwork at the secondary side of the KNXnet/IP Router and should therefore also have the same Subnetwork Address as the KNXnet/IP Router, in line with the IAs of the Subnetwork at the secondary side.

3. The procedure below firstly handles the PID_ADDITIONAL_INDIVIDUAL_ADDRESSES in the KNXnet/IP Server Object and then only PID_INDIVIDUAL_ADDRESS in the Device Object. It is recommended practice that the KNXnet/IP Server that supports more than one KNXnet/IP Tunnelling connection, does not use the own Individual Address (PID_INDIVIDUAL_ADDRESS) for KNXnet/IP Tunnelling.

3.2.3 Procedure

3.2.3.1 General

It is the intention that the KNXnet/IP Tunnelling Client executes the steps in the clause 3.2.3 one after the other.

3.2.3.2 Step 1: Establish a KNXnet/IP Tunnelling connection to the device and get the IA used for the KNXnet/IP Tunnelling Connection

This is the KNXnet/IP Tunnelling Connection of which the IA will be changed. If the KNXnet/IP Tunnelling connection is already open, then this step shall be skipped and the existing KNXnet/IP Tunnelling connection shall be used.

```
        /* The KNXnet/IP Client builds up a KNXnet/IP Tunnelling connection */
        /* to the KNXnet/IP Tunnelling Server. */
        /* The KNXnet/IP Tunnelling Server returns the used IA as part of the CRD in the response. */
DMP_KNXnet/IP_Connect(dmp_HPAIControlEndpoint, dmp_HPAIDataEndpoint,
        dmp_CRI = TUNNEL_CONNECTION, dmp_CommChannelID, dmp_Status, dmp_HPAIServerDataEndpoint,
        dmp_CRD)
```

If **IA$_{req}$ = dmp_CRD.IA** then the procedure can be halted here: the used IA is the requested IA$_{req}$.

3.2.3.3 Read operations

3.2.3.3.1 Step 2: Build up a KNXnet/IP Management Connection

```
        /* The KNXnet/IP Client builds up a KNXnet/IP Device Management connection */
        /* to the KNXnet/IP Tunnelling Server. */
DMP_KNXnet/IP_Connect(dmp_HPAIControlEndpoint, dmp_HPAIDataEndpoint,
        dmp_CRI = DEVICE_MGT_CONNECTION, dmp_CommChannelID, dmp_Status,
        dmp_HPAIServerDataEndpoint, dmp_CRD)
```

3.2.3.3.2 Step 3: Read the additional IAs of the KNXnet/IP Tunnelling Server

NOTE 3 The Additional Individual Addresses IAadditional[] are read firstly and then only the IA$_{device}$ is read. This fits best to the models B and C in Figure 1, in which the KNXnet/IP Tunnelling Server does not use its own device Individual Address as KNXnet/IP Tunnelling Address.

```
DMP_InterfaceObjectRead_IP(mpp_Obj.Type = KNXnet/IP Parameter Object, mpp_Obj.Inst = 1,
        mpp_Prop.ID = PID_ADDITIONAL_INDIVIDUAL_ADDRESSES, IAadditional[] = mpp_Prop.Value,
        IAadditionalNr = mpp_Prop.CurrentNr)
```

NOTE 4 In this procedure, it is assumed that there is only one instance of the KNXnet/IP Parameter Object. Implementations with more than one instance of this Interface Object, e.g. with more than one IP connection, are not yet considered in this specification.

The resulting information is the list of additional Individual Addresses, IAadditional[], that is maintained by this KNXnet/IP Tunnelling Server. This list shall be queried for the presence of two values.

1. The CRD.IA used for the open KNXnet/IP Tunnelling connection.
2. The IA$_{req}$ requested by the ETS user.

1. Search CRD.IA in IAadditional[]

The CRD.IA should be searched in this list IAadditional[]. There are three possible results.

a. CRD.IA is found exactly once in IAadditional[].

b. CRD.IA is found more than once IAadditional[].

c. CRD.IA is not found at all in IAadditional[].

a. CRD.IA is found once at the index CRD.IAindex

ETS shall write IA$_{req}$ in PID_ADDITIONAL_INDIVIDUAL_ADDRESSES at this index CRD.IAindex according 3.2.3.5.2.

This shall be regardless of whether it is in 3.2.3.3.3 found that CRD.IA would also match IA$_{device}$.

b. CRD.IA is found more than once

If CRD.IA is found more than once in PID_ADDITIONAL_INDIVIDUAL_ADDRESSES[] then ETS shall write IA$_{req}$ at the lowest index where it is found in PID_ADDITIONAL_-INDIVIDUAL_ADDRESSES at this index CRD.IAindex according 3.2.3.5.2.

This shall be regardless of whether it is in 3.2.3.3.3 found that CRD.IA would also match IA$_{device}$.

NOTE 5 This can be the case if a newly installed KNXnet/IP Tunnelling Server has all array elements of PID_ADDITIONAL_INDIVIDUAL_ADDRESSES[] for instance filled with FFFFh.

NOTE 6 It may be that the index CRD.IA is not the index of the entry for CRD.IA that is taken by the KNXnet/IP Tunnelling Server for this connection. This is only possible in the very rare case that one or more other KNXnet/IP Tunnelling Clients would manage the Additional Individual Addresses in parallel and uses the same value for its IA_{req}.

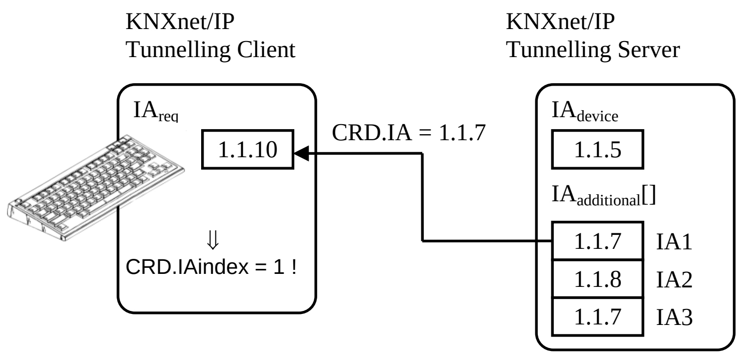

Figure 2 – Selection of CRD.IAindex

c. CRD.IA is not found in IAadditional[]

In this case, CRD.IA should match IA_{device}. IA_{device} shall be read according 3.2.3.3.3.

2. Check for the possible presence of IA_{req} in IAadditional[]

- If also IA_{req} is in this list, then it is possible that IA_{req} is already used by another, open KNXnet/IP Tunnelling connection. It cannot be discovered whether or not this is the case. Therefore, IA_{req} cannot be used as IA for this KNXnet/IP Tunnelling Connection.

  ETS should report that the chosen IA_{req} may currently not be possible and request the ETS user to select a different IA_{req}. The procedure is exit here.

- If IA_{req} is not present in this list, then it still may be that IA_{req} equals the IA_{device} of the KNXnet/IP Tunnelling Server device. Therefore, IA_{device} shall in any case be read.

3.2.3.3.3 Step 4: Read the IA_{device} of the KNXnet/IP Tunnelling Server

```
        /* Read PID_KNX_INDIVIDUAL_ADDRESS in the KNXnet/IP Parameter Object */
        /* of the KNXnet/IP Tunnelling Server. */
DMP_InterfaceObjectRead_IP(mpp_Obj.Type = KNXnet/IP Parameter Object, mpp_Obj.Inst = 1
        mpp_Prop.ID = PID_KNX_INDIVIDUAL_ADDRESS, IAdevice = mpp_Prop.Value, 1)
```

The response shall be subject to a sequence of tests.

1. Check if IA_{device} equals CRD.IA

If IA_{device} = CRD.IA then the device uses its own IA as Tunnelling Address for this KNXnet/IP Connection.

NOTE 7 This is the typical case for a KNXnet/IP Tunnelling Server that supports only a single KNXnet/IP Tunnelling Connection. Obviously, this can however also occur for a KNXnet/IP Tunnelling Server that supports more than one KNXnet/IP Tunnelling connection.

If all below tests prove positive, the assignment shall then be done by writing PID_KNX_-INDIVIDUAL_ADDRESS as specified in 3.2.3.5.3.

2. Check if IA_{device} equals IA_{req}

If thus $IA_{device} \neq$ CRD.IA but however $IA_{device} = IA_{req}$ then the device already has the IA requested for the KNXnet/IP Tunnelling connection, but did not assign it to this KNXnet/IP Tunnelling Client, e.g. as another KNXnet/IP Tunnelling connection using IA_{device} already stands.

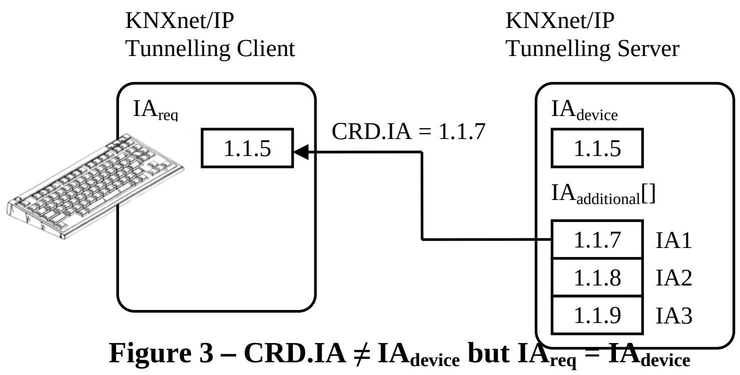

Figure 3 – CRD.IA ≠ IA_{device} but IA_{req} = IA_{device}

ETS should report to the ETS user that the requested IA_{req} is used as the own IA of the KNXnet/IP Tunnelling Server and is not used for the KNXnet/IP Tunnelling Connection. It can propose to change IA_{req}.

The procedure is exit here.

3.2.3.4 Tests

3.2.3.4.1 Step 5: Compare with IA_{device}

IA_{req} shall be compared with IA_{device}. If the Subnetwork Address is different, then ETS should signal this to the ETS user and request the user to select a different IA_{req} with the same Subnetwork Address as in IA_{device}. This is certainly relevant if the KNXnet/IP Tunnelling Server is hosted in a KNXnet/IP Router.

NOTE 8 This should only be a warning. It should always be possible to use a KNXnet/IP Tunnelling Address (IA_{req}) with an SNA that is not identical to the SNA that is not identical to the SNA of the KNXnet/IP Tunnelling Server (IA_{device}).

NOTE 9 This is only a consistency measure when assuming a new KNXnet/IP Tunnelling Address. If the Subnetwork Address of the KNXnet/IP Tunnelling Server changes later on, there is no requirement towards the KNXnet/IP Tunnelling that it “corrects” any of its KNXnet/IP Tunnelling IAs.

3.2.3.4.2 Step 6: Read suitable SNAs from the Couplers

Another, possible, additional way to check whether an IA_{req} contains a valid Subnetwork Address is by reading it from the Router.

```
    /* This procedure will always work independently of any IA used by the KNXnet/IP Tunnelling Server */
    /* as it uses broadcast communication mode. */
NM_NetworkParameter_Read_R(hop_count_type_req = 0, object_type = Device Object,
    PID = PID_SUBNET_ADDR, test_info = 00h, comm_mode_res = broadcast, hop_count_type_res = 0,
    result_data[])
```

The Couplers that are with their primary or secondary side in the same segment as the KNXnet/IP Tunnelling Server will reply at that side. From the number and values of result_data[], the topological position of the KNXnet/IP Server can be concluded and it can be concluded if IA_{req} would be a suitable IA. Please refer to “SNA Read” in [2].

The KNXnet/IP Tunnelling Client should have a time-out on the bus [3)] for collecting any answer of at least 3 s.

[3)] This is the time on the bus between the transmission of the request and the reception of the answer. the KNXnet/IP Tunnelling Client should take into account the times needed for Datagrams on IP to travel forth and back to the KNXnet/IP Tunnelling Server.

3.2.3.4.3 Step 7: Check whether IA_{req} is free

If IA_{req} is not found in the KNXnet/IP Tunnelling Server, then it has to be tested whether IA_{req} is an acceptable IA.

NOTE 10 The following procedure may use the actual standing KNXnet/IP Tunnelling Connection. This procedure will not be possible however if that connection is not Tunnelling on Data Link Layer; ETS should give a warning that this test about uniqueness of the IA_{req} cannot be performed.

```
        /* The KNXnet/IP Tunnelling Client uses its KNXnet/IP Tunnelling Connection */
        /* to call the next procedure. */
NM_IndividualAddress_Check(IA_test = IAreq)
```

If this results positive, then it is sure that IA_{req} is used in the Subnetwork and cannot be chosen. ETS should signal this to the ETS user and request the user to select a different IA_{req}. The Configuration Procedure is stopped here and restarted from the beginning if the ETS user selects a different IA_{req}.

If this results negative, then it is not guaranteed that IA_{req} is really a free IA, namely if IA_{req} does not have a suitable SNA, the present Routers do not support SNA-reading or have incorrect SNAs. With the preceding tests, this risk should be minimal.

3.2.3.5 Step 8: Assignment of IA_{req}

3.2.3.5.1 General

IA_{req} shall be written in the KNXnet/IP Tunnelling Server by one of the below procedures. Which procedure shall be applied depends on where CRD.IA has been found. This is given in the read operations in 3.2.3.3.

3.2.3.5.2 Assignment via PID_KNX_ADDITIONAL_INDIVIDUAL_ADDRESSES[]

PID_ADDITIONAL_INDIVIDUAL_ADDRESSES does not need to be sorted. In the following, the IA_{req} can thus be written at the Property Value array element where CRD.IA has been found before, this is CRD.IAindex, without further re-arranging (resorting) the remaining Property Value array elements.

If CRD.IA is found at multiple indexes, then IA_{req} shall be changed at the lowest index.

```
        /* Change the Property Array Element of PID_ADDITIONAL_INDIVIDUAL_ADDRESSES[] /*
        /* where CRD.IA is found above: at CRD.IAindex */
DMP_InterfaceObjectWrite_IP(mpp_Obj.Type = KNXnet/IP Parameter Object, mpp_Obj.Inst = 1,
        mpp_Prop.ID = PID_ADDITIONAL_INDIVIDUAL_ADDRESSES, mpp_Prop.Value = IAreq,
        mpp_Prop.StartIndex = CRD.IAindex, mpp_Prop.NrOfElem = 1, mpp_ErrorCode)
```

The KNXnet/IP Tunnelling Server shall apply and use the newly assigned IAreq immediately, without requiring a restart. The M_PropWrite.con contained in this procedure DMP_InterfaceObjectWrite_IP shall be the confirmation that the KNXnet/IP Tunnelling Address has changed and is used.

3.2.3.5.3 Assignment via PID_INDIVIDUAL_ADDRESS

NOTE 11 The below procedure will be used if the KNXnet/IP Tunnelling Server has used its own IA as Tunnelling Address. A KNXnet/IP Router shall not use its own IA for KNXnet/IP Tunnelling connections (see Figure 1-C), in the below it is not necessary to check whether the KNXnet/IP Tunnelling Server is in a KNXnet/IP Router.

```
        /* Change the IA of the KNXnet/IP Tunnelling Server. /*
DMP_InterfaceObjectWrite_IP(mpp_Obj.Type = KNXnet/IP Parameter Object, mpp_Obj.Inst = 1,
                mpp_Prop.ID = PID_KNX_INDIVIDUAL_ADDRESS, mpp_Prop.Value = IAreq,
                mpp_Prop.StartIndex = 1, mpp_Prop.NrOfElem = 1, mpp_ErrorCode)
```

3.2.3.6 Closing the procedure

The modified Tunnelling Address shall apply immediately.

The Management Client shall close the established KNXnet/IP Device Management connection if it is no longer needed.

Further requirements

- ETS shall not execute any restart after any of these Procedures.

3.3 Handling of E_NO_MORE_UNIQUE_CONNECTIONS by the KNXnet/IP Tunnelling Client

If a KNXnet/IP Tunnelling Servers returns a KNXnet/IP Tunnelling CONNECT_REQUEST with a CONNECT_RESPONSE with the error constant E_NO_MORE_UNIQUE_CONNECTIONS then this means that the KNXnet/IP Tunnelling Server could provide a connection (in contrast to error constant E_NO_MORE_CONNECTIONS) if only the KNXnet/IP Tunnelling Address that would be assigned to the connection would be unique. Then the client (ETS) shall behave as follows.

1. Build up a KNXnet/IP Management Connection as specified in 3.2.3.3.1.
2. Read the additional IAs (PID_ADDITIONAL_INDIVIDUAL_ADDRESS) of the KNXnet/IP Tunnelling Server as specified in 3.2.3.3.2.
3. In the results, search for the first occurrence of a duplicate KNXnet/IP Tunnelling Address.
   - If at least one double entry is found, let this be the address $IA_{multiple}$ at index $IA_{muliple}$.index.

     NOTE 12 There may be more than one IA that appears multiple times in the list. Only the first found IA will however be handled.
   - If no double entry is found, continue at 4b.
4. Report to the ETS user that the KNXnet/IP Tunnelling device does not accept the KNXnet/IP Tunnelling connection because the Individual Address $IA_{multiple}$ is used multiple times within the KNXnet/IP Tunnelling device. The user must then provide a different value for that Individual Address. Let this address be IA_{unique}. ETS shall evaluate IA_{unique} according the tests specified in 3.2.3.4.
5. Using KNXnet/IP Device Management, replace the second instance of $IA_{multiple}$ with IA_{unique} with the method as specified in 3.2.3.5.2.

   NOTE 13 Only the second instance is changed and not the first one or any further instance: if the error E_NO_MORE_UNIQUE_CONNECTIONS is returned, it is very likely that the first instance is used by an open KNXnet/IP Tunnelling Connection, of which the IA should not be changed (unless by the KNXnet/IP Tunnelling Client that owns that connection).
6. Retry establishing a KNXnet/IP Tunnelling Connection to the KNXnet/IP Tunnelling Server. This may have the following results.
   - If this retry is successful, this KNXnet/IP Tunnelling Connection can be used and the procedure is closed here.
   - If this retry returns again in E_NO_MORE_UNIQUE_CONNECTIONS, then the above procedure shall be repeated from 2.
   - If this retry results in other errors (e.g. E_NO_MORE_CONNECTIONS) then these shall be handled as specified for these errors.

PID_ADDITIONAL_INDIVIDUAL_ADDRESS does not contain any double entry

4b. Read the IA_{device} of the KNXnet/IP Tunnelling Server as specified in 3.2.3.3.3.

NOTE 14 IA_{device} should match at least one entry in the before read PID_ADDITIONAL_INDIVIDUAL_-ADDRESS. If not, this would indicate a KNXnet/IP Tunnelling Server error.

5b. If IA_{device} equals FFFFh, then ETS shall request the ETS user to enter a new value for IA_{device}. Let this address be IA_{unique}. ETS shall only accept a value complying with the following.

- It shall not conflict with any other used Individual Address in the installation.
- It shall not be present in the list of additional Individual Addresses PID_ADDITIONAL_-INDIVIDUAL_ADDRESS.

ETS shall evaluate IA_{unique} according the tests specified in 3.2.3.4.2 and 3.2.3.4.3.

ETS shall change the IA of the KNXnet/IP Tunnelling Server, by setting PID_INDIVI-DUAL_ADDRESS to the value IA_{unique} as specified in 3.2.3.5.3.

ETS shall retry establishing a KNXnet/IP Tunnelling Connection to the KNXnet/IP Tunnelling Server. This may have the following results.

- If this retry is successful, this KNXnet/IP Tunnelling Connection can be used and the procedure is closed here.
- If this retry returns again in E_NO_MORE_UNIQUE_CONNECTIONS, then the above procedure shall be repeated from 2.
- If this retry results in other errors (e.g. E_NO_MORE_CONNECTIONS) then these shall be handled as specified for these errors.

6b. If IA_{device} differs from FFFFh, all entries in PID_ADDITIONAL_INDIVIDUAL_ADDRESSES that equal PID_INDIVIDUAL_ADDRESS shall be set to FFFFh, using the procedure specified in 3.2.3.5.2.

ETS shall retry establishing a KNXnet/IP Tunnelling Connection to the KNXnet/IP Tunnelling Server. This may have the following results.

- If this retry is successful, this KNXnet/IP Tunnelling Connection can be used and the procedure is closed here.
- If this retry returns again in E_NO_MORE_UNIQUE_CONNECTIONS, then the above procedure shall be repeated from 2.
- If this retry results in other errors (e.g. E_NO_MORE_CONNECTIONS) then these shall be handled as specified for these errors.

4 Frame structures

4.1 Introduction

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

4.2 Common constants

Refer to [4] for a list of valid KNXnet/IP common constants.

4.3 Common error codes

Refer to [4] for a list of valid KNXnet/IP common error codes.

4.4 KNX telegram tunnelling

KNXnet/IP Tunnelling shall be initiated by establishing a KNXnet/IP Tunnelling connection from the KNXnet/IP Client, e.g. ETS, to the KNXnet/IP Server. Refer to [5] for details.

4.4.1 KNXnet/IP services

Table 1 lists all valid KNXnet/IP service types for KNXnet/IP Tunnelling.

Table 1 – Tunnelling KNXnet/IP service type identifiers

Service name	Code	V.	Description
TUNNELLING_REQUEST	0420h	1	Used for sending and receiving single KNX frames between KNXnet/IP Client and - Server.
TUNNELLING_ACK	0421h	1	Sent by a KNXnet/IP Client or – Server to confirm the reception of the TUNNELLING_REQUEST.

4.4.2 Connection Type

The connection type value for the connection type TUNNEL_CONNECTION shall be 04h.

Refer to [5] for more details.

4.4.3 Connection Request Information (CRI)

The Connection Request Information (CRI) contains the requested Tunnelling KNX layer.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   TUNNEL_CONNECTION           |
|   (1 octet = 04h)             |   (1 octet = 04h)             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   KNX Layer                   |   reserved                    |
|   (1 octet)                   |   (1 octet)                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 4 – Tunnelling CRI binary format

Table 2 – Tunnelling KNX layers

Constant name	Value	V.	Description
TUNNEL_LINKLAYER	02h	1	Establish a Data Link Layer tunnel to the KNX network.
TUNNEL_RAW	04h	1	Establish a raw tunnel to the KNX network.
TUNNEL_BUSMONITOR	80h	1	Establish a Busmonitor tunnel to the KNX network.

Table 3 – Tunnelling CONNECT_ACK error codes

Error constant	Value	V.	Description
E_NO_ERROR	00h	1	The message was received successfully.
E_TUNNELLING_LAYER	29h	1	The requested tunnelling layer is not supported by the KNXnet/IP Server device.

4.4.4 Connection Response Data Block (CRD)

The Connection Response Data Block (CRD) shall contain the KNX Individual Address assigned to this KNXnet/IP Tunnelling connection.

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   TUNNEL_CONNECTION           |
|   (1 octet = 04h)             |   (1 octet = 04h)             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   KNX Individual Address                                      |
|   (2 Octets)                                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 5 – Tunnelling CRI binary format

4.4.5 Connection header

```
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   Communication Channel ID    |
|   (1 octet)                   |   (1 octet)                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  |   Sequence Counter            |   reserved                    |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
```

Figure 6 – Tunnelling connection header binary format

4.4.6 TUNNELLING_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-+
 |   TUNNELLING_REQUEST                                          |
|   (0420h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(Connection Header) +                |
|       sizeof(cEMI Frame)                                      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

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

                           cEMI frame
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Message Code                |   Additional Info Length      |
|   (1 octet = 08h)             |   (1 octet)                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Additional Information                                      |
|   (optional, variable length)                                 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Service Information                                         |
|   (variable length)                                           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 7 – TUNNELLING_REQUEST frame binary format

4.4.7 TUNNELLING_ACK

```
                         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-+
   |   TUNNELLING_ACK                                          |
|   (0421h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(Connection Header)                  |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

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

Figure 8 – TUNNELLING_ACK frame binary format

5 Binary examples of KNXnet/IP frames

5.1 TUNNELLING_REQUEST

```
        +-------------------------------+ - - - - KNXnet/IP header - - - -
   1    |              06h              |    header size
        +-------------------------------+
   2    |              10h              |    protocol version
        +-------------------------------+
   3    |              04h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0420h
   4    |              20h              | /
        +-------------------------------+
   5    |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, L+12 octets
   6    |            L+0Ch              | /
        +-------------------------------+ - - - - connection header - - - -
   7    |              06h              |    structure length of connection header
        +-------------------------------+
   8    |              15h              |    communication channel ID, e.g. 21
        +-------------------------------+
   9    |              00h              |    sequence counter
        +-------------------------------+
  10    |              00h              |    reserved
        +-------------------------------+ - - - - cEMI frame - - - -
  11    |              11h              |    message code (e.g. L_Data.req message)
        +-------------------------------+
  12    |              00h              |    additional information (none)
        +-------------------------------+
  13    |              ...              | \
        +-------------------------------+ |
  14    |              ...              |  > Service Information (L bytes)
        +-------------------------------+ |
 L+12   |              ...              | /
        +-------------------------------+
```

Figure 9 – TUNNELLING_REQUEST frame binary format: example

5.2 TUNNELLING_ACK

```
        +-------------------------------+ - - - - KNXnet/IP header - - - -
   1    |              06h              |    header size
        +-------------------------------+
   2    |              10h              |    protocol version
        +-------------------------------+
   3    |              04h              | \
        +- - - - - - - - - - - - - - - -+  > service type identifier 0421h
   4    |              21h              | /
        +-------------------------------+
   5    |              00h              | \
        +- - - - - - - - - - - - - - - -+  > total length, 10 octets
   6    |              0Ah              | /
        +-------------------------------+ - - - - connection header - - - -
   7    |              04h              |    structure length of connection header
        +-------------------------------+
   8    |              15h              |    communication channel ID, e.g. 21
        +-------------------------------+
   9    |              00h              |    sequence counter
        +-------------------------------+
  10    |              00h              |    status, e.g. 00h (NO_ERROR)
        +-------------------------------+
```

Figure 10 – TUNNELLING_ACK frame binary format: example

6 Certification

6.1 Introduction

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

6.2 Support matrix

Service name	sent from ... to ...	implementation is
TUNNELLING_REQUEST	Client → Server (L_Data.req, M_Reset.req) Server → Client (L_Data.con, L_Data.ind)	M
TUNNELLING_REQUEST	Client → Server (L_Raw.req) Server → Client (L_Raw.con, L_Raw.ind, L_Busmon.ind)	O
TUNNELLING_ACK	Client → Server Server → Client	M

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