System Specifications

Management

Resources

3

5

1

Summary

This document specifies the elements that are available in the network and in the device to the Management Client and that shall be managed by this Management Client.

Version 01.09.03 is a KNX Approved Standard.

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

Document updates

Version	Date	Modifications
1.0	2001.12.13	Integration of comments from FV, CSG feedback.
1.2	2007.01.05	• Supplement 18 “Network Resources” integration started and completed.
1.2	2007.01.07	• Supplement 15 “Easy Common Parts”, requirements on Individual Address, etc. integrated. • Started list of references below.
1.2	2007.01.15	• AN038 “Function Services”: PID_APPLICATION
1.2		• AN029 “DPSU on TP1” integrated.
1.2	2007.11.07	• AN031 “Coupler Resources” integrated.
1.2	2007.11.12	• AN033 “Common EMI” integrated.
1.2	2007.11.23	• Added: OptionReg,
1.2	2008.01.16	• AN043 “LTE on RF”: PID_SERIAL_NR_TABLE
1.2	2008.01.17	• AN044 “RF Specification Complements”: PID_OBJECTADDRESS
1.2	2008.02.13	• Added indication about inverse coding of bit 2 in PID_SERVICE_CONTROL, acc. KSG conclusion of 2007.12.04-05.
1.2	2008.04.28	• AN066 “cEMI adaptations” integrated.
1.2	2008.04.29	• AN032 “Easy Resources” integrated.
1.2	2008.05.05	• AN067 “APCI_ServiceInformation_Indication_Write” integrated.
1.2	2008.05.07	• AN071 “Link services and DMA access” integrated.
1.2	2008.07.14	• AN060 “Bidirectional Battery dirven devices” integrated.
1.2	2008.08.01	• AN081 “2-level topology”
1.2	2008.08.05	• AN090 “Discovery of long frame range”
1.2	2008.08.05	• AN091 “Telegram rate limitation for System B” integrated. • Introduction of Resource “Group telegram rate limitation”
1.2	2008.08.05	• AN094 “DD0 for devices with and without direct bus connection” integrated. • Added mask version 0AFDh.
1.2	2008.08.06	• AN095 “Non-selective Data Link Layer acknowledge” integrated
1.2	2008.08.07	• AN078 “cEMI Extensions for RF and BiBat” integrated
1.2	2008.08.11	• AN102 “TP1 Bridges” integrated
1.2		• Merged Group Object Association Table Realisation Type 7 (System 300) IN Realisation Type 6 (System B).
1.2	2008.08.12	• AN103 “Structure of PID_MGT_DESCRIPTOR_01” integrated. • AN104 “Global Property PID_OBJECT_INDEX” integrated. • AN101 “File Transfer Protocol” integrated.
1.2	2008.12.02	• Added PID_DEVICE_DESCRIPTOR in Device Object and in KNXnet/IP Parameter Object.
1.2	2009.01.08	• Preparation of the Approved Standard v1.2.
1.2	2009.01.21	• Removed PID_DEVICE_DESCRIPTOR from the KNXnet/IP Parameter Object.
1.2	2009.01.22	• Added indication that the Properties of the KNXnet/IP Parameter Object are specified in Chapter 3/8/3 “Management” for completeness.
1.2	2009.02.04	• Preparation for inclusion in the KNX Specifications v2.0.
1.2	2009.03.31	• Editorial update of Load- and Run State Machines
1.2	2009.06.23	• Editorial update in view of inclusion in the KNX Specifications v2.0.
1.2.01	2009.11.24	• Editorial updates
1.2.02	2010.10.22	• AN127 “Master Reset” integrated
1.3.00	2010.12.23	• AN115 “Mask 5705h” integrated
01.04.00	2012.09.10	• Revision of the integration of AN127 “Master Reset” according the updated AN127 “Master Reset” v05.
01.05.00	2012.10.22	• AN149 “New Load State LoadCompleting” integrated.

Version	Date	Modifications
01.05.01	2013.03.19	• Accepted all changes in the document. • Removed use restrictions of PID_HARDWARE_TYPE to mask 0705h and 0025h. • PID_PROGRAM_VERSION: Property Name and PDT corrected. • PID_TABLE_REFERENCE: Property Name corrected.
01.05.02	2013.06.13	• PID_HARDWARE_TYPE: removed reference to V06 “Profiles” and eased the reading of the use of MSB 00h. • PID_TABLE_REFERENCE: PDT corrected. • PID_MCB_TABLE: corrected multiple appearances of PDT_GENERIC_07[] to be PDT_GENERIC_08[]. • Multiple editorial corrections.
01.06.00	2013.07.18	• AN146 “Ctrl-Mode for KNX RF Multi” integrated.
01.07.00	2013.07.19	• AN148 “PB-Mode for KNX RF Multi” integrated.
01.07.01	2013.07.19	• AN151 “cEMI AddInfo for KNX RF Multi and new Properties” integrated.
01.07.02	2013.07.22	• Editorial review.
01.08.00	2013.09.03	• AN137 “Configuration Signature” integrated.
01.08.01	2013.10.18	• Editorial review.
01.09.01	2013.10.22	• AN153 “Mask 0912h Property based configuration” integrated. • Transfer of Coupler Resources of masks 0910h and 0911h from Part 9/3 “Couplers”.
01.09.02	2013.10.28	Editorial updates for the publication of KNX Specifications 2.1.
01.09.03	2013.12.10	Final editorial review in view of publications of the KNX Specifications v2.1.

References

[01] Chapter 3/1/2 “Glossary”
[02] Chapter 3/2/2 “Communication Medium TP1”
[03] Chapter 3/3/2 “Data Link Layer General”
[04] Chapter 3/3/3 “Network Layer”
[05] Chapter 3/3/4 “Transport Layer”
[06] Chapter 3/3/7 “Application Layer”
[07] Chapter 3/4/1 “Application Interface Layer”
[08] Chapter 3/5/2 “Management Procedures”
[09] Chapter 3/5/3 “Configuration Procedures”
[10] Chapter 3/6/2 “Physical External Interface”
[11] Chapter 3/6/3 “External Message Interface”
[12] Chapter 3/7/2 “Datapoint Types”
[13] Chapter 3/7/3 “Standard Identifier Tables”
[14] Chapter 3/8/3 “Management” (KNXnet/IP)
[15] Volume 6 “Profiles”
[16] Volume 7 “Application Descriptions”
[17] Chapter 8/2/2 “TP1 Physical and Link Layer Tests”
[18] Chapter 9/4/1 “BCUs and BIMs”

Filename:	03_05_01 Resources v01.09.03 AS.docx
Version:	01.09.03
Status:	Approved Standard
Savedate:	2013.12.10
Number of pages:	226

Contents

1 Introduction

1.1 Scope

To allow for KNX devices to function properly in the network in which they are installed, all KNX devices shall allow for being configured, from the minimum level up to a more or less advanced configuration functionality. This document targets on specifying data structures that shall be used for configuring KNX devices. These data structures are called *Resources*.

The standard KNX system communication stack already provides the necessary functionality for accessing Resources. This functionality consists either of dedicated services in the Application Layer (see [06]) or the generic architecture of the Interface Objects.

This document specifies the existing standard Resources of the KNX system. For every Resource it specifies the configuration functionality that can be managed by it, the data structure of the Resource, how the Resource shall be managed by a Management Client and how the Management Server – typically the device – shall interpret the Resource's data.

The Management Procedures shall target on managing these Resources.

1.2 Definition

1.2.1 Contained information

Resources are these data in a device that contain information relevant to the device configuration.

The information contained in Resources serves for

- device identification and discovery

  EXAMPLES serial number, device descriptor, manufacturer, device type, …

- addressing and binding

  EXAMPLES Individual Address, Group Address Table, Group Object Association Table, Group Object Table, …

- application configuration

  EXAMPLES application code, application parameters, …

This list is not exclusive.

1.2.2 Access rights

This data can be

- read-only, this is, provided by the device for information to a Management Client, but not changeable by a Management Client, or
- read-write, this is, a Management Client can set the Resource data in the device in order to influence its behaviour.

1.2.3 Access ways

A Resource can be accessed by

- a Management Client, remote

  The Resource is accessed (read and/or write) from the bus by a Management Client, using dedicated services.

- a Management Client, local

  The Resource is accessed via the PEI (see [10]) and the EMI (see [11]).

This method is relevant when a Management Client, typically the S-Mode Tool, configures the device to which it is directly connected via the PEI.

It is however also relevant in case of a dual-processor device. These devices are based on standard BAU serving as host for the bus access for an external processor. The external processor shall use the same access methods to the Resources hosted by the BAU as specified for the local Management Client.

- a Management Server, internal

  This specifies how the device shall access and interpret the information contained by the Resource.

This differentiation in access ways is reflected in the individual Resource descriptions in the following subclauses.

1.2.4 Types of Resources

In this document a differentiation is made between *Network Resources* and *Device Resources*.

Network Resources are these Resources of which the possible data range is shared with other devices. The data set in the Network Resources of different devices shall either

- match, to make these devices work together

  EXAMPLES Domain Address, Subnetwork Address…

- differ, to avoid communication conflicts

  EXAMPLES Individual Address (Subnetwork Address + Device Address), KNX Serial Number …

Device Resources are these Resources of which the possible data solely serves for the configuration of the device in which they are contained. The contained data has no relation to the data of Resources in other devices; these Resources do not (directly) influence the relationship between this device and any other device.

1.3 Specifying Resources through their management interface

The specification of Resources given below is actually the specification of their management interface. This means that the identifiers (maybe even location identifiers like “memory addresses”), encoding and formats given, specify the content of the management messages to control (i.e. read, write or provide) the (elements of) a Resource. *The way this is mapped to internal memory locations (including bit order etc.) is not defined and left entirely to the implementation.*

In other words: the specification lays down how these Resources are seen and handled by a Management Client (either local or remote). As such, they are essentially platform-independent.

This specification lays no requirements on the internal storage and modelling in the Management Client. The Management Client shall be able to map between its internal storage style and the modelling required by the Resource specifications below.

1.4 Abbreviations

Please refer to [01] for the abbreviations used in this document.

2 Realisation types of Resources

2.1 Common elements of Resource definitions

A Realisation Type fixes a particular standard implementation for a Resource (interface). This implies certain choices are made, for example concerning:

- the length of identifiers (e.g. SAPs), which influences the required storage space as well as the allowable range;

  NOTE 1- The term Service Access Point (SAP) is defined in ISO/IEC 74981-1 "The Basic Model" as "The point at which (N)-services are provided by the (N)-entity to an (N+1)-entity".

- sorting of “lookup” Resources, to speed up their evaluation, etc.
- (location) identifiers, usually corresponding to a (historical) memory mapped implementation.

In the clauses of this document, a *common structure* is used to describe a realisation type of a Resource.

1. *Format*

   This clause specifies the image of the Resource realisation type as it is visible by its interfaces: this can be a memory mapped structure or an Interface Object or a Property of an Interface Object. It is up to the implementer of the Realisation Type to make this image match with his data.

2. *Location*

   This clause specifies the way to access the realisation type: this can be an address for memory mapped Resource types or Object Types and Property Identifiers for Resources implemented as (part of) Interface Objects.

3. *Usage by the Management Server*

   This clause specifies how the Resource shall be evaluated by the services that base on it.

4. *Usage by the Management Client*

   This clause specifies how any Management Client shall manage the realisation type of the Resource. A differentiation is made in function of the access way to the actual Resource:

4.1 Remote access	This specifies how the Resources shall be managed when being accessed via the bus.
4.2 Local access	This specifies how the Resources shall be managed when being accessed via the PEI.
4.3 Internal Access	This specifies how an internal application shall access this Resource if it takes direct access to it.

2.2 Allowed usage of realisation types

Some realisation types may be restricted for future use, as “not recommended” or “not allowed” in the Resource container (server). In this case, they are specified to allow the implementation of backward compatible Management Clients.

To accommodate a limited, well-defined set of implementations, for several Resources there may be more than one realisation type defined.

One complete system implementation should provide a sufficiently homogenous set of Resource management interfaces. For this reason, realisation types of different Resources shall not be combined at will. Rather, the allowable combinations are laid down in the Profile specifications (see [15]).

2.3 E-Mode Resources

2.3.1 Differences between S-Mode Resources and E-Mode Resources

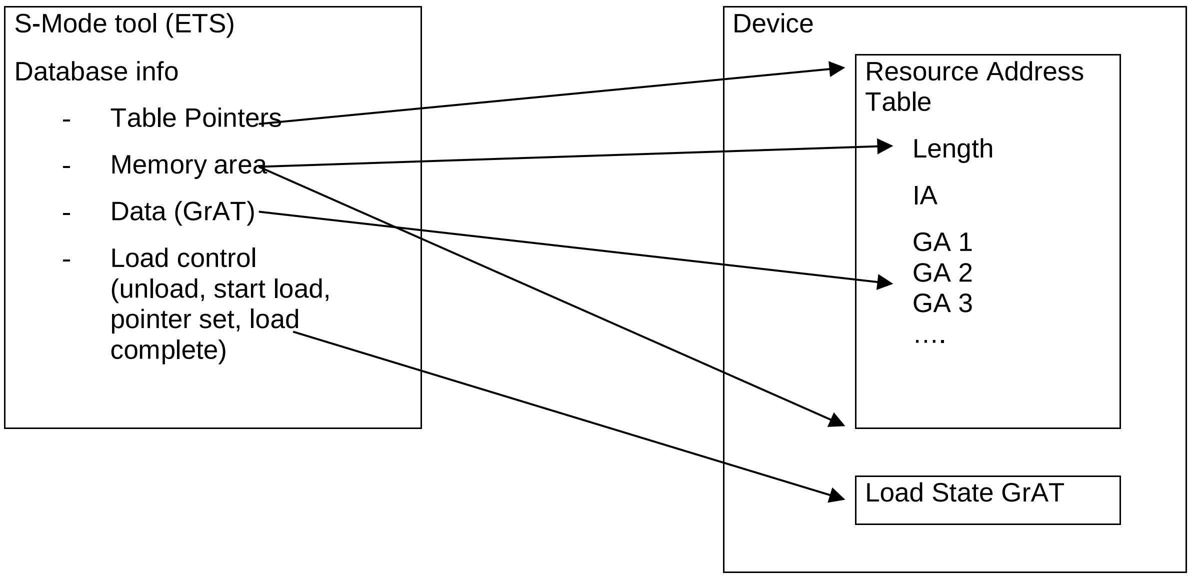
The S-Mode Resources and the E-Mode Resources are very similar, but there is a distinction in handling of table data, pointers and procedures. For the access inside the device through the device management there is no difference, in fact the device itself can be both an E-Mode device as well an S-Mode device.

2.3.2 S-Mode Resources

The S-Mode Resources are designed for a download into an S-Mode device by the S-Mode Management Client, typically a computer based software. This procedure requires the information for table data, for the pointers, the available memory Resource inside the device in the tools’s database.

With this information the Resource can be handled e.g. a Group Address Table can be created, filled in and started.

EXAMPLE Group Address Table System mode

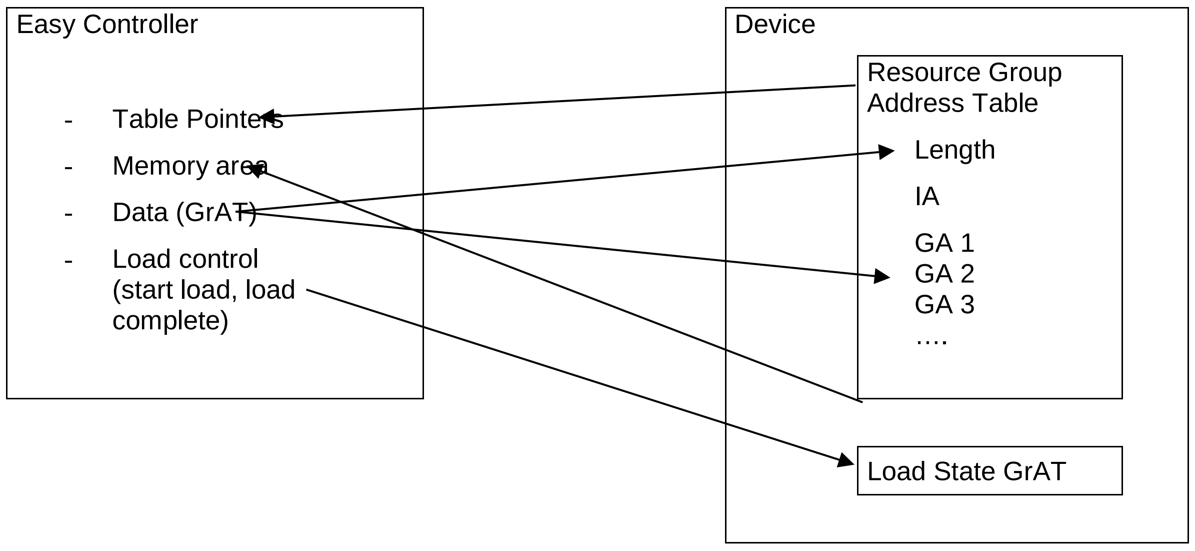


2.3.3 E-Mode Resources

The E-Mode Resources for the Ctrl-Mode are designed for writing and reading in a preloaded Ctrl-Mode device. The Resource can be read and written like a handling read - modify - write back.

EXAMPLE The content of an existing Group Address Table can be changed. The Group Addresses are generated by the controller; the pointers and memory information are fetched from the E-Mode device. The (existing) GrAT is stopped, changed and then restarted.

EXAMPLE Group Address Table System mode



2.3.4 Resource data integrity in case of dual access by Link Services and memory mapped access

In devices providing memory mapped access and Link Services, the same Resources shall be shared (Group Address Table, Group Association Table and Group Object table). A device may provide both mechanisms but must ensure Resources' consistency. For this, Resources shall not be accessed with both mechanisms at one given time.

In this condition, an E-Mode Controller may use both mechanisms for link management. It does not have to check the device state before downloading via memory mapped access or Link Services.

3 Network Resources

3.1 Overview

In the KNX System, the network addresses are encoded on a 16 bit field, providing an addressable range of 65 535 addresses.

The usage of the address space for the Individual Addresses is subject to the structured topology that is designed for the network. The format of the Individual Address is specified in [03]. The *Subnetwork Address* reflects the position of the Subnetwork to which the device is connected in the topology.

The address space available for devices - the *Device Address space* - allows for 256 different Device Addresses on a Subnetwork. Table 2 indicates the allocation of Device Addresses per Subnetwork.

The address space available for groups of objects or devices - the *Group Address space* - is 16 bit wide, allowing for up to 65 535 of addresses on the whole network. It is recommended to assign the Group Addresses only once per network to avoid potential inconsistent network behaviour and to simplify the integrity tests for the Group Address assignments that are made.

Table 4 indicates the recommended partitioning of the Group Address space. The partitioning is particularly important when KNX installations make use of the different Configuration Modes.

3.2 Domain Addresses

3.2.1 Definition

If the medium is an open medium and if this has to be shared with instances of other Subnetworks, a Domain Address (DoA) is required as an identifier to logically separate the Subnetworks from each other.

Each Domain Address shall identify a logical Subnetwork on an open medium. *On some open media the address space may be reduced or practically avoided (e.g. direct line-of-sight links with IR).*

Alternative solutions than the Domain Address are possible to limit communication on open media to one or more Subnetworks of a given KNX installation. For instance, communication can be limited to devices with a known KNX Serial Number. Please refer to the specification of the various communication media in Part 3/2 "Communication Media" for the options per medium.

3.2.2 Receiving frames

A device shall always receive frames that are sent using the Broadcast Domain Address, whether it has gained an Individual Address for its runtime operation (through installation procedures, registration or contention) or not. The Destination Address – Group – or Individual - shall be evaluated and be considered for acceptance.

To avoid abuse of the broadcast Domain Address it is recommended to use this address only in a controlled (guarded) situation. Therefore, the device's management processes will decide on the acceptance and execution of the message.

3.2.3 DoA Realisation Type 1 – DoA on 2 octets

The Domain Address shall be encoded on a 16 bit field, providing over 65 535 possible Domain Addresses,

Table 1 - Domain Address space

Domain Address	Usage
FFFFh	DoA for use as non-divided open medium
FFFEh	reserved Domain Addresses
FFF0h	
FFEFh	free Domain Address space
00FFh	(e.g. boundary of a reduced DoA space)
0001h	
0000h	Broadcast Domain Address

The Domain Addresses space is subdivided in four segments with a specific or recommended use.

1. A first segment is the single Domain Address 0000h that is exclusively reserved as a Broadcast Domain Address. This address shall be supported by devices that use the open medium for Domain Address management.
2. The second segment of Domain Addresses ranges from 0001h to FFEFh. The usage of the Domain Address in this range shall be subject to prior Domain Address Acquisition procedures to create logical Subnetworks on an open medium.
3. The third segment of Domain Addresses, ranging from FFF0h to FFFEh, is reserved for future extensions. These Domain Addresses shall not be used.

The fourth segment is the single Domain Address FFFFh that shall be reserved for simple or small configurations and environments where the division of the open medium in logical Subnetworks is not required. This Domain Address shall not be used by the Domain Address Acquisition procedures.

NOTE 2 The support of this Domain Address will actually enable a system to operate without any assignment of a Domain Address by a central Management Client, or self acquisition, or a Domain Address Server

In case the Domain Address space is smaller than the indicated 65 535, the Domain Address space shall start from the broadcast Domain Address and consecutively to higher Domain Addresses. The Broadcast Domain Address shall always be the first Domain Address in the available address space.

EXAMPLE In Table 1 the address 00FFh is the upper value of a reduced Domain Address space, used to reduce the hardware demands on devices using E-Mode, or open media with a rather low density of Domain Addresses.

3.2.4 DoA Realisation Type 2 – DoA on 6 octets

If unconfigured (ex-factory) the Domain Address is void and shall have the default value 000000000000h.

LTE devices shall not use the default Domain Address for runtime communication.

3.3 Individual Addresses

The allocation of the Device Addresses - as part of the Individual Addresses - is given in Table 2.

The Subnetwork Address is subject to the hierarchical network topology and shall as such be assigned by a tool, the installer (end-user) or one of the Network Management procedures.

The table of the Device Addresses, and in particular the pre-assigned addresses, shall be applied to every Subnetwork.

The addresses 00h and FFh are reserved Device Addresses.

- Address 00h shall be used by the Router on any given Subnetwork.
- Address FFh shall be used as a default Device Address for any device that has not got assigned a Device Address.

The reserved Device Addresses shall be respected by any Configuration Mode.

Table 2 - Device Address space

Device Address	Usage
FFh	Unregistered Devices
FEh	free
FDh	free
05h	free
04h	free
03h	free
02h	free
01h	free
00h	Router

If there are no means to determine the local Subnetwork Address, the medium dependent default Subnetwork Address shall be used.

Table 3 - Medium dependent default Subnetwork Addresses

Configuration Mode	Default Subnetwork Address	Medium	Resulting medium dependent default IA
• **S-Mode**	FFh	All media	FFFFh
• **All E-Modes**	01h	reserved	01FFh
	02h	TP 1	02FFh
	03h	reserved	03FFh
	04h	PL 110	04FFh
	05h	RF	05FFh
	06h to FFh	reserved	not applicable

The procedure for assigning the Individual Address to a device depends on the Configuration Mode(s) supported by the device. Please refer to [09]. These Configuration Procedures use the Management Procedures NM_DomainAndIndividualAddresss_Write2, NM_IndividualAddress_Write, NM_IndividualAddress_SerialNumber_Write, NM_IndividualAddress_SerialNumber_Write2 or other as specified in [08].

3.4 Group Addresses

3.4.1 Definition

If Group Addresses used in one Subnetwork would be re-used for different purposes in one or more other Subnetworks, then this can lead to consistency problems and loss of integrity of the applications. Therefore, Group Addresses shall be unique on the whole network.

The Group Addresses are assigned as 16 bit wide identifiers, allowing for 65 535 Group Addresses [1]. This is specified in [03].

3.4.2 Group Address space

3.4.2.1 Ranges

The Group Address space is subdivided in four ranges with a specific or recommended use. Table 4 reflects this subdivision.

A first range is the single Group Address 0000h that is exclusively reserved for the Broadcast Address that shall be exclusively used in broadcast communication mode.

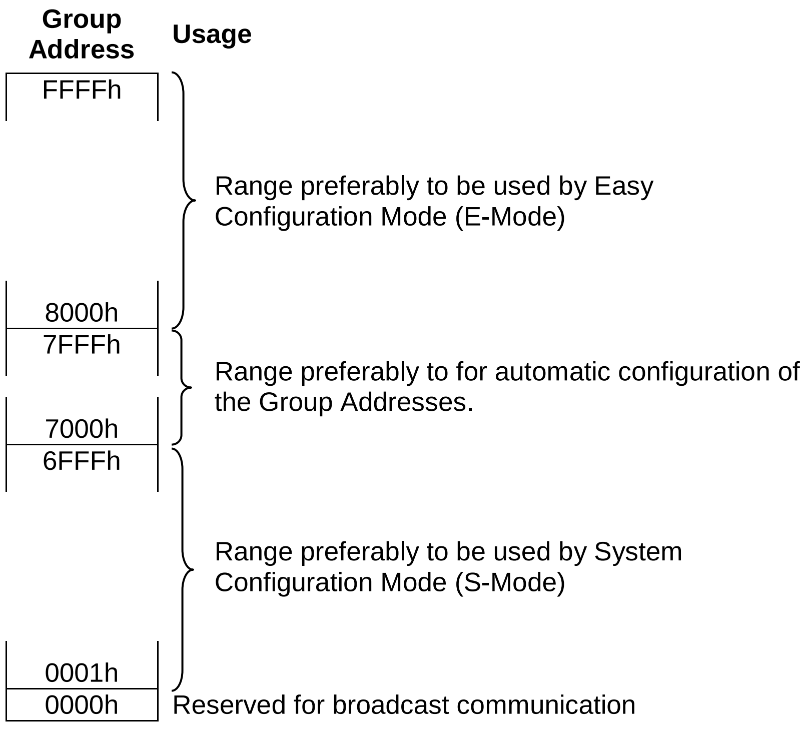
The further ranges are made only to reduce the risk of conflicts in multicast communication mode with Group Addresses when the KNX system comprises more than one Configuration Mode. It is recommended to respect these ranges; however, per installation it may be decided to extend the range, foreseen that the end user does not introduce inconsistencies.

The range of 0001h to 6FFFh is preferably used for the System Configuration mode (S-Mode).

The range of 7000h to 7FFFh is preferably used for automatic configuration of the Group Addresses.

The range of 8000h to FFFFh is preferably used for the Easy Configuration mode (E-Mode).

Table 4 - Group Address space


Group Address
Usage
FFFFh
Range preferably to be used by Easy Configuration Mode (E-Mode)
8000h
7FFFh
Range preferably to for automatic configuration of the Group Addresses.
7000h
6FFFh
Range preferably to be used by System Configuration Mode (S-Mode)
0001h
0000h
Reserved for broadcast communication

[1] The Extended Frame Format allows for a larger range. Please refer to Part 10/1 “LTE Specification”.

3.4.2.2 Range 8000h to FFFFh - Range preferably used by E-Mode

Range 8000h to FFFFh is further subdivided, defining ranges to be used for Functions of Common Interest, Group Address Ex-factory, Predefined addresses … This is specified in Table 5 below.

Table 5 - Subdivision of Group Address segment to be used by E-Mode

Start address	End address	Number of addresses	Usage
FF00h	FFFFh	256	• Reserved for specific system usage
F400h	FEFFh	2 816	• Reserved for future extensions
F000h	F3FFh	1 024	• Group address ex-factory • Group Addresses for Functions of Common Interest See 3.4.2.3 below.
E000h	EFFFh	4 096	• Reserved for future extensions
C000h	DFFFh	8 192	• Free area with Group Address checking See 3.4.2.4 below.
8000h	BFFFh	16 384	• reserved See 3.4.2.5 below.

The KNX System specifies a specific procedure to check if a Group Address is already in use or not. This is specified as the procedure NM_GroupAddress_Scan in [08]. It is mandatory to execute this procedure with any Group Address one wants to assign in the range C000h to DFFFh.

In the segment for Auto Configuration Mode, the Group Addresses 7000h to 700Fh are reserved for network management specific use.

3.4.2.3 Range F000h to F3FFh

Predefined Group Addresses for functions of common interest and Group Address ex-factory share the same address range F000h to F3FFh] according the following assignment scheme.

Group address	Usage
F000h	Group addresses ex-factory
	↓
	↑
F3FFh	Group address for Functions of Common Interest

Table 6 specifies the reserved Group Addresses that have been assigned already.

Table 6 – Assigned Group Addresses for functions of common interest and Group Addresses ex-factory

Group Address	Type	Usage
…		
F3F3h	FOCI	current electricity tariff
…		
F3F8h	FOCI	ChangeOverStatusWater (DPT_Heat/Cool, 1.100)
F3F9h	FOCI	Outside Temperature (DPT_Value_Temp, 9.001)
F3FAh	FOCI	Technical Alarm "AlarmInfo"
F3FBh	FOCI	Technical Alarm "AlarmText"
F3FCh	reserved	reserved
F3FDh	reserved	reserved
F3FEh	FOCI	current date and time
F3FFh	reserved	reserved

Other Group Addresses than the above in this range, which have not yet been assigned, shall not be used and are under the control of KNX Association.

3.4.2.4 Range C000h to DFFFh

Before using a Group Address in the range C000h to DFFFh a Management Client shall ensure that this Group Address is free before assigning it (either by check, or by systematic scanning of links in devices, assuming it is possible for all devices)

Warning

Once some change has been made with the tool in this address space, the tool has to be reused when making subsequent changes in the installation.

3.4.2.5 Range 8000h to BFFFh

This range is not used.

4 Device Resources

4.1 Device Descriptors

4.1.1 Overview

The following Descriptor Types are specified

- Device Descriptor Type 0

  Abbreviation: DD0

  Synonym: “mask version”

and

- Device Descriptor Type 2

  Abbreviation: DD2

  This Device Descriptor Type contains also an application description.

Any KNX device shall support at least one Descriptor Type.

The types 1 and 3 to 63 are reserved for future use by KNX Association and shall not be used in implementations.

NOTE 3 Device Descriptor Type 0 is designed for use in S-Mode. Existing S-Mode devices may respond to an A_DeviceDescriptor_Read in which the Descriptor Type differs from 0 with an A_DeviceDescriptor_Response-PDU containing the Device Descriptor formatted according DD0 but the field Descriptor Type as in the request, thus different from 0. The conclusion on the real contained Descriptor Type can then only be based on the length of the field device_descriptor. Therefore, no new Descriptor Type should have a Device Descriptor format of 2 octets.

4.1.2 Device Descriptor Type 0 (mask version)

4.1.2.1 Device Descriptor Type 0 Format

The Device Descriptor Type 0 shall have the format as specified in Figure 1.

Mask Type (8 bit)		Firmware Version (8 bit)	
MMMM	TTTT	VVVV	SSSS
Medium Type	Firmware Type	Version	Subcode

Figure 1 – Device Descriptor Type 0 format

4.1.2.2 Device Descriptor Type 0 assigned values

Table 7 – KNX Association approved mask versions for devices

Mask version	Mask Type Medium Type	Mask Type Firmware Type	**Firmware Version**	**Subcode**	**Medium**	**Device Profile**
0010h	0h	0h		0	TP1	BCU 1, System 1
0011h	0h	0h		1	TP1	BCU 1, System 1
0012h	0h	0h		2	TP1	BCU 1, System 1
0013h	0h	0h	1h	3	TP1	BCU 1, System 1
0020h	0h	0h		0	TP1	BCU 2, System 2
0021h	0h	0h		1	TP1	BCU 2, System 2
0025h	0h	0h	2h	5h	TP1	BCU 2, System 2
0300h	0h	3h	0h	0	TP1	System 300
0700h	0h	7h		0	TP1	BIM M112
0701h	0h	7h		1	TP1	BIM M112
0705h	0h	7h	0h	5	TP1	BIM M112
07B0h	0h	7h	Bh	0h	TP1	System B
0810h	0h	8h		0	TP1	IR-Decoder
0811h	0h	8h		1	TP1	IR-Decoder
0910h	0h	9h		0	TP1	Coupler 1.0
0911h	0h	9h		1	TP1	Coupler 1.1
0912h	0h	9h		2	TP1	Coupler 1.2
091Ah	0h	9h	1h	tbd	IP (prim) TP1 (sec)	KNXnet/IP Router
0AFDh	0h	Ah	Fh	Dh	TP1	none (see V06 "Profiles")
0AFEh	0h	Ah	Fh	Eh	TP1	none (see V06 "Profiles")
1012h	1h	0h		2	PL110	BCU 1
1013h	1h	0h		3	PL110	BCU 1
17B0h	1h	7h	Bh	0	PL110	System B
1900h	1h	9h		0	TP1-PL110	Media Coupler PL-TP
2010h	2h	0h		0	RF	Bi-directional devices
2110h	2h	1h		0	RF	Uni-directional devices
3012h	3h	0h		2	TP0	BCU 1
4012h	4h	0h		2	PL132	BCU 1
5705h	5h	7h	0h	5h	KNX IP	System 7

4.1.3 Device Descriptor Type 2

4.1.3.1 Device Descriptor Type 2 format

The Device Descriptor Type 2 shall have the format as specified in Figure 2.

Octet 0 (MSB)	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6	Octet 7	Octet 8	Octet 9	Octet 10	Octet 11	Octet 12	Octet 13 (LSB)
Application Manufacturer		Device Type		Version	Misc. + LT-Base	Channel Info 1		Channel Info 2		Channel Info 3		Channel Info 4	

Figure 2 – Device Descriptor Type 2 format

Field Name	Offset (Octets)	Size	Coding
Application Manufacturer	0 and 1	16 bit	Manufacturer code. Value binary encoded.
Device Type	2 and 3	16 bit	Manufacturer specific Device Type
Version	4	8 bit	Version of manufacturer specific Device Type
Miscellaneous	5: bits 7 and 6	2 bit	00b: link management services not supported 01b: link management services supported 10b: undefined 11b: undefined
LT_Base	5: bits 5 to 0	6 bit	Base value for logical tags, derived from Local Selector Value 3Fh means no Selector active
Channel Info 1	6 and 7	16 bit	Number and type of channels, first type
Channel Info 2	8 and 9	16 bit	Number and type of channels, second type
Channel Info 3	10 and 11	16 bit	Number and type of channels, third type
Channel Info 4	12 and 13	16 bit	Number and type of channels, fourth type

All fields shall always be provided, even if unused.

4.1.3.1.1 Application Manufacturer

This field shall contain the manufacturer code of the manufacturer that produces the device. This manufacturer code is assigned by KNX Association.

4.1.3.1.2 Device Type

This field shall contain a manufacturer specific Device Type.

4.1.3.1.3 Version

This field shall contain the version of the manufacturer specific Device Type.

4.1.3.1.4 Miscellaneous

This field shall contain a 2-bit value that shall indicate whether or not Network Management procedures based on the A_Link_Read-service and the A_Link_Write-service are supported by the device.

4.1.3.1.5 Logical Tag Base (LT_Base)

This field shall contain the current value of the 6 bit Logical Tag Base.

It shall be calculated from the local selector value, according to a unique correspondence table. Please refer to [09] Appendix 1 “Logical Tag Installer Procedures - Rules”.

For multichannel devices, the value is aligned to the closest lowest authorised value, according to the following rule:

- 2-channels devices: (LT_Base modulo 2) = 0
- 3- or 4-channels devices: (LT_Base modulo 4) = 0

Specific Values (unchanged in case of multichannel devices):

- 3Fh: no active local selector
- 3Eh: general, all zones
- 3Dh: reserved
- 3Ch: reserved

4.1.3.1.6 Channel Information

The channel information shall provide the functional description of a given channel.

The Channel Information shall be encoded on 16 bit:

- bits 15 to 13:	3 bit:	number of channels of following channel code The possible number of channels of a given type implemented in a device can be 1 to 8 (value 0 means 1 channel).
- bits 12 to 0:	13 bit:	channel code

If a channel does not exist the Channel Code shall be 0000h. For unknown channels the Channel Code shall be set to 0001h.

The Channel and corresponding Channel Codes are described in for every application in the appropriate Chapters in [16].

Some Channel Code values are reserved as specified in Table 8.

Table 8 – Reserved Channel Code values

Channel Code	Use
0000h	unused (then no more valid data afterwards)
0001h	descriptor not provided through this service This value is reserved by KNX Association for future extensions. It shall not be used.
1FF0h	Localisation Channel for 1 to 8 Channels
1FF1h	Localisation Channel for 1 to 16 Channels
1FF2h	Localisation Channel for 1 to 24 Channels
1FF3h	This Channel Code shall not be used.
1FF4h	This channel code shall be used by LTE-Mode devices
1FF5h to 1FFFh	system reserved

4.1.3.1.7 Descriptor Type 1 and 3 to 7

These Device Descriptor Types are reserved by KNX Association for future extensions and shall not be used.

4.2 Interface Object Type independent Properties

4.2.1 PID_OBJECT_TYPE (PID = 1)

- Property name: Interface Object Type
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: DPT_PropDataType (DPT_ID = 7.010)

This Property shall have the value of the Object Type of the Interface Object. This Property is always mandatory and shall always be readable.

EXAMPLE In the Device Object, this Property shall have the value 0000h.

4.2.2 PID_OBJECT_NAME (PID = 2)

- Property name: Interface Object Name
- Property Datatype: PDT_UNSIGNED_CHAR[]
- Datapoint Type: None.

This Property shall include the name of the Interface Object.

In case this Property is implemented in the Device Object (Object Type = 0), then this Property shall be interpreted as the name of the device.

This Property is always optional.

This is only a descriptive value. The string shall be encoded in ASCII / ISO 8859-1 as an array of characters (unsigned char[]; the max. number of elements shall be fixed in the device).

The access rights of this Property depend on the implementation.

4.2.5 PID_LOAD_STATE_CONTROL (PID = 5)

- Property name: Load Control
- Property Datatype: PDT_CONTROL
- Datapoint Type: None.

This Property shall control the Load State Machine of the hosting Interface Object. A write-access to this Property shall be equal to an event for the Load State Machine and a read-access to this Property shall provide the current state of the Load State Machine. The response value shall always be one octet and include the current state of the Load State Machine. The write value shall always be 10 octets. Its first octet shall be the event for the Load State Machine. The following octets shall be additional information for the Load State Machine.

The Load State Machine is specified in clause 4.17.

This Property is mandatory if the application is loadable. If this Property does not exist the Application Program is not loadable.

4.2.6 PID_RUN_STATE_CONTROL (PID = 6)

- Property name: Run Control
- Property Datatype: PDT_CONTROL [84)]
- Datapoint Type: None.

This Property shall control the execution of the executable part of the hosting Interface Object. A write access to this Property shall be equal to an external event for the Run State Machine and a read-access to this Property shall provide the current state of the Run State Machine.

The Run State Machine is specified in clause 4.18.

4.2.7 PID_TABLE_REFERENCE (PID = 7)

- Property name: Table Reference
- Property Datatype: PDT_UNSIGNED_LONG
- Datapoint Type: None.

A Management Client shall use the value of this Property as reference for memory mapped access to this Resource. The Management Server (device) shall set this reference when the Management Client allocates the memory space for this Resource. This shall be a read-only Property, which shall be used by the Management Client for memory mapped access (using A_Memory_Write and A_Memory_Read) to the Resource.

The address shall be a 4 octet pointer to an absolute memory space location within the device. Due to the fact that A_UserMemory_Write and A_UserMemory_Read services are limited to a 20 bit address range the reference pointer shall not exceed this range. The most significant 12 bit shall be reserved for future use. A_Memory_Read; A_Memory_Write, A_UserMemory_Read and A_UserMemory_Write shall operate on the same physical address space (except that A_Memory_Read and A_Memory_Write shall be restricted to the lower 64 kb, while A_UserMemory_Read and A_UserMemory_Write shall be able to address the full range of 1 Mb).

If the Load State of the related Resource changes to ‘unloaded’ then the Management Server shall set the value to zero. When the Management Client has successfully allocated memory then the value shall be set to the absolute memory location from where the Resource will be downloaded. When memory allocation is not successful then the value shall be set to zero.

4.2.8 PID_SERVICE_CONTROL (PID = 8)

- Property name: Service Control
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

Individual Address Write Enable is enhanced as specified in AN057 “System B”.
The difference between which Profile has to support which bits is made in V06 “Profiles”.

PID_SERVICE_CONTROL shall be a permanent control field for the device. It shall be structured as specified in Table 9.

Table 9 – PID_SERVICE_CONTROL

Octet	Bit		Encoding (d = default value)
Low Octet	0	Reserved. Shall be 0 [2].	0
	1	Reserved. Shall be 0 [2].	0
	2	Individual Address Write Enable	0 = disable (d) 1 = enable
	3	Reserved	-
	4	Reserved	-
	5	Reserved	-
	6	Reserved	-
	7	Reserved	-
High octet	8	Application Interface Layer Services on EMI Disable	0 = enable 1 = disable
	9	Link Layer Services on EMI Disable	0 = enable 1 = disable
	10	Network Layer Services on EMI Disable	0 = enable 1 = disable
	11	Transport Layer Group Services on EMI Disable	0 = enable 1 = disable
	12	Switch Service-Services on EMI Disable	0 = enable 1 = disable
	13	Transport Layer Connection Oriented Services on EMI Disable	0 = enable 1 = disable
	14	Application Layer Services on EMI Disable	0 = enable 1 = disable
	15	Management Services on EMI Disable	0 = enable 1 = disable

Octet 1 of the Property Service Control shall only be valid for devices with an EMI. For devices without EMI all services shall always be disabled.

- Bit 2 IndividualAddress_Write:

  This bit shall control the possibility to set the Individual Address via programming mode or KNX Serial Number services. If this bit is cleared, it shall not be possible to change the Individual Address of the device.

  Mask 0021h implementations shall have inverse coding for this bit:

2	Individual Address Write Enable	0 = enable 1 = disable

NOTE 4 The specification of which KNX Profile shall support which fields in PID_SERVICE_CONTROL is given in Volume 6 “Profiles”.

[2] In PID_SERVICE_CONTROL, bit 0 and bit 1shall be cleared. They are used in legacy implementations for the control of the abandoned service A_ServiceInformation_Indication-Write, as follows:

Bit		Encoding
0	User Stopped_ServiceInfo Enable Enables the generation of an A_ServiceInformation_Indication_Write report of the device in case of unexpected stop of the user program.	0 = disable 1 = enable
1	Received own Individual Address Service Info enable Enables the generation of a ServiceInformation Report of the device in case of reception of a frame with the own Individual Address as source address.	0 = disable 1 = enable

4.2.9 PID_FIRMWARE_REVISION (PID = 9)

- Property name: Firmware Revision
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall contain the revision number of the firmware of the object in which it is contained. Values shall start with 00h.

This Property is encoded as a single octet unsigned character.

This information can also be stored in PID_VERSION (PID = 25), which is encoded as DPT_Version. Volume 6 "Profiles" specifies which of the given Profile(s) shall be implemented.

4.2.10 PID_SERVICES_SUPPORTED (PID = 10)

- Property name: Services Supported
- Property Datatype: None.
- Datapoint Type: None.

This Property shall contain information about the services that are supported by this device.

4.2.11 PID_SERIAL_NUMBER (PID = 11)

- Property name: Serial Number
- Property Datatype: PDT_GENERIC_06
- Datapoint Type: DPT_SerNum (DPT_ID = 221.001)

This Property shall contain the KNX Serial Number of the device in which it is contained. PID_SERIAL_NUMBER is the Realisation Type 2 of the Resource KNX Serial Number as specified in 4.16.3.

4.2.12 PID_MANUFACTURER_ID (PID = 12)

- Property name: Manufacturer Identifier
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

Code assigned by the KNX Association to identify the manufacturer of the device.

4.2.13 PID_PROGRAM_VERSION (PID = 13)

- Property name: Program Version
- Property Datatype: PDT_GENERIC_05
- Datapoint Type: None.

This Property shall contain the application version of the object in which it is contained.

NOTE 5 The Property Identifier of this Property was formerly named PID_APPLICATION_VERSION.

4.2.14 PID_DEVICE_CONTROL (PID = 14)

• Property name:	Device Control
• Property Datatype:	PDT_BITSET8 (alt.: PDT_GENERIC_01)
• Datapoint Type:	DPT_Device_Control (DPT_ID = 21.002)

4.2.14.1 Abstract Resource definition

Property Device Control shall be a temporary control field for the device.

4.2.14.2 Format

Table 10 – Device Control

Bit	Usage
0	User stopped
1	Individual Address duplication
2	Verify Mode On
3	Safe State On
4 … 7	Reserved

4.2.14.3 Usage by the Management Client

A Management Client can reset the value by writing 0 to this Property.

4.2.14.4 Usage by the Management Server (device)

The device shall set and clear the fields as specified in the clauses below.

The device shall set the value of this Property to 00h on start-up.

4.2.14.5 User stopped

4.2.14.5.1 Encoding

0	The user program is running.
1	The user program is stopped.

4.2.14.5.2 Usage by the Management Server (device)

This bit shall be set by the device if the user program stopped for internal reason (e.g. wrong PEI-Type).

4.2.14.6 Individual Address Duplication

4.2.14.6.1 Encoding

0	No frame has been received with the own Individual Address as Source Address
1	At least one frame has been received with the own Individual Address as Source Address

4.2.14.6.2 Usage by the Management Server (device)

This bit shall be set by the device if a frame has been received with the own Individual Address as Source Address of the received frame.

4.2.14.7 Verify Mode Control

4.2.14.7.1 Abstract Resource Definition

Verify Mode controls the reaction of the Management Server for handling of the following Application Layer services primitives as specified in [06]:

- an A_Memory_Write.ind primitive, as specified for the A_Memory_Write-service, or
- an A_MemoryBit_Write.ind primitive, as specified for the A_MemoryBit_Write-service, or
- an A_Memory_Write.ind primitive, as specified for the A_Memory_Write-service, or
- an A_UserMemoryBit_Write.ind primitive, as specified for the A_UserMemoryBit_Write-service, or
- an A_Write_Routing_Table.ind primitive, as specified for the A_Write_Routing_Table-service, or
- an A_Write_Router_Memory.ind primitive, as specified for the A_Write_Router_Memory-service.

Verify Mode shall per default be disabled in the Management Server. A Management Client can enable Verify Mode via the Verify Mode Control. Verify Mode can only be enabled by the Management Client if there is a Transport Layer connection to the Management Server. Verify Mode shall automatically be disabled by the Management Client when the Transport Layer connection is closed.

The Verify Mode Control is a device resource to control the Verify Mode of the Management Client functionality in the device.

4.2.14.7.2 Encoding

Used by:	mask 0020h, 0021h, 0701h, 0705h
	System B, System 300
	mask 091Ah

1 bit

0	Verify mode is disabled
1	Verify mode is enabled

4.2.14.7.3 Usage by the Management Server (device)

The value of Verify Mode Control shall per default be 0 ("disabled"). When a Transport Layer connection to the device is closed, the Management Server shall immediately and automatically disable Verify Mode and clear the Verify Mode Control bit.

4.2.14.7.4 Usage by the Management Client

When opening a Transport Layer connection to the Management Server, the Management Client shall assume the Verify Mode Control to be cleared; it shall actively set the Verify Mode Control.

When closing a Transport Layer connection, the Management Client may assume the Verify Mode Control to be automatically cleared and thus not have to clear it explicitly.

4.2.14.8 Safe State Control

4.2.14.8.1 Abstract Resource definition

Safe State denotes a state of the device in which no user-supplied part shall be executed.

The implementation of the Safe State is optional.

4.2.14.8.2 Encoding

1 bit

0	The device is not in Safe State.
1	The device is in Safe State.

4.2.14.8.3 Usage by the Management Server (device)

The device may enter in Safe State through manufacturer specific procedures.

The change back to normal mode shall only be possible through reset.

4.2.14.8.4 Usage by the Management Client

The Safe State Control shall be used to switch to Safe State via an A_PropertyValue_Write. If the Safe State is not implemented bit 3 shall be reserved and set to 0.

4.2.15 PID_ORDER_INFO (PID = 15)

• Property name:	Order Info
• Property Datatype:	PDT_GENERIC_10
• Datapoint Type:	None.

This Property shall contain manufacturer specific Order Information (PDT_GENERIC_10).

4.2.16 PID_PEI_TYPE (PID = 16)

• Property name:	PEI Type
• Property Datatype:	PDT_UNSIGNED_CHAR
• Datapoint Type:	None.

This Property shall contain the currently connected PEI-Type (“Hardware Type”). This value shall regularly be updated, e.g. on every cycle through the system stack. It shall be evaluated as specified in [10]. In devices without PEI the value shall be zero.

4.2.17 PID_PORT_CONFIGURATION (PID = 17)

• Property name:	PortADDR
• Property Datatype:	PDT_UNSIGNED_CHAR
• Datapoint Type:	None.

This Property shall include the settings of the hardware configuration of a device (e.g. on BCU PortA direction control register 0 = input; 1 = output). The value of this Property is manufacturer specific.

A Management Client shall read this value and compare it to a value stored for this device in its repository (database). Only if both values equal, the Management Client may continue with the configuration of the device’s application, such as download of the user application code or its parameters. The manufacturer shall take care setting the value of this Property to the value required by the hardware.

This Property is not usable for LTE-Mode devices.

4.2.18 PID_POLL_GROUP_SETTINGS (PID = 18)

- Property name: Pollgroup Settings
- Property Datatype: PDT_POLL_GROUP_SETTINGS
- Datapoint Type: None.

This Property value shall be composed of:

- the Polling Group Address: 2 octets
- the Polling Disable bit: 1 bit

0	Polling shall be enabled.
1	Poling shall be disabled.

- the Polling Slot Number: 4 bit
- This field shall contain the polling slot number in the range 0 to 15.

Polling Group Address																Disable	Reserved			Polling Slot Nr.			
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	D	0	0	0	S	S	S	S

4.2.19 PID_MANUFACTURER_DATA (PID = 19)

- Property name: Manufacturer Data
- Property Datatype: Variable
  - PDT_GENERIC_04
- Datapoint Type: Not applicable.

This shall be a Property with manufacturer specific device information (e.g. manufacture date …). The Property Datatype of this Property is not defined. It depends on the data a manufacturer wants to be handled there. Therefore the Property Datatype of the Property must be read out via A_Property-Description_Read before data can be read. This requires the implementation of full Interface Objects.

4.2.21 PID_DESCRIPTION (PID = 21)

- Property name: Description
- Property Datatype: PDT_UNSIGNED_CHAR[]
- Datapoint Type: None.

This Property shall include additional descriptive information for the device and is optional in any case.

The descriptive information shall be encoded in ASCII / ISO 8859-1 as an array of characters (unsigned char[]; the maximum number of elements shall be fixed in the device). The access rights of this Property depend on the implementation.

4.2.23 PID_TABLE (PID = 23)

- Property name: Table
- Property Datatype: None.
- Datapoint Type: None.

The Property Table is used in various Interface Objects for the storage of data in table format. The format (2 octets, 4 octets…) and contents (Group Addresses, Associations…) depend on the host object. Please refer therefore to the use of PID_TABLE in the Interface Object specifications further in this document.

4.2.25 PID_VERSION (PID = 25)

- Property name: Version
- Property Datatype: PDT_VERSION (alt.: PDT_GENERIC_02)
- Datapoint Type: DPT_Version (DPT_ID = 217.001)

This Property shall specify version information related to the Interface Object in which it resides. Version information is encoded using the new standard DPT_Version.

In case of the Device Object PID_VERSION shall be the version information for the device itself.

EXAMPLE Revision number of the firmware.

The interpretation of the device version information is product specific. The entry point for interpretation of any data with versioning is the Device Object Property PID_PRODUCT_ID (PID = 55).

4.2.27 PID_MCB_TABLE (PID = 27)

- Property name: Memory Control Table
- Property Datatype: PDT_GENERIC_08[]
- Datapoint Type: None

This Property shall define and control the subsegmentation of a relating memory mapped Resource and its checksum.

Writing the Memory Control Block Table shall only have effect in the Load State 'Loading'.

Table 11 - Memory Control Block (MCB) Table

Nr. of elements				
Segment Size 1 [4 octets]	CRC Control Byte [1 octet]	Read Access 1 [4 bit]	Write Access 1 [4 bit]	CRC [2 octets]

4.2.27.1.1 CRC Control Byte

Table 12 – CRC Control Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	0 : CRC is always valid 1 : Contents of protected memory area may change

If the CRC Control Byte, Bit 0 is set to zero this shall indicate that the protected memory area will never be changed after download. So the CRC shall always be the calculated checksum of the protected memory area.

If the CRC Control Byte, Bit 0 is set to 1 this shall indicate that the protected memory area can be changed after the load process by an application.

4.2.27.1.2 CRC

The CRC checksum calculation shall provide a higher consistency while using differential download.

The CRC shall be calculated properly by the Management Server (device) on the transition from the load state 'Loading' to the load state 'Loaded' according to the segment's memory content right from the beginning. Its value shall be valid in the load state 'Loaded' only.

The 16 bit CRC shall be calculated according to the CRC16-CCITT specification.

The CRC of the Memory Control Block Table Property is a CRC16-CCITT with the following parameters:

Width = 16 bit

Truncated polynomial = 1021h

Initial value = FFFFh

Input date is NOT reflected.

Output CRC is NOT reflected.

No XOR is performed on the output CRC.

EXAMPLE The correct CRC16-CCITT of the string ‘123456789’ is E5CCh.

4.2.28 PID_ERROR_CODE (PID = 28)

• Property name:	Error code
• Property Datatype:	PDT_ENUM8 (alt. PDT_UNSIGNED_CHAR)
• Datapoint Type:	DPT_ErrorClass_System (20.011)

This Property shall indicate the last error that occurred before the load state ‘Error’ is set. When the load state changes from ‘Error’ to a different state then the Error Code shall be set to ‘0’ (no error).

For the error code description see DPT_ErrorClass_System in [12]. This list defines all specified error codes. A manufacturer may implement a subset of these. The definition of new error codes is an ongoing process.

4.2.29 PID_OBJECT_INDEX (PID = 29)

• Property name:	Object Index
• Property Datatype:	PDT_UNSIGNED_CHAR
• Datapoint Type:	DPT_Value_1_Ucount

This Property shall contain the local Object Index of the Interface Object in which it is located.

The Property PID_OBJECT_INDEX is optional in any Profile.

The Property PID_OBJECT_INDEX is optional in any Interface Object.

The Property PID_OBJECT_INDEX shall always be read-only.

Usage by the Management Server (device)

The Property PID_OBJECT_INDEX shall be readable via LTE Read/Response Services (A_GroupPropValue_Read-PDU and A_GroupPropValue_Response-PDU). If the Property is implemented in the Interface Object, then the Property shall be readable through all LTE zones used by the Interface Object.

The other LTE-services (A_GroupPropValue_InfoReport and A_GroupPropValue_Write) are not supported for this Property.

The Property PID_OBJECT_INDEX shall be readable, as every Property, also through the point-to-point communication mode Property Services (A_PropertyValue_Read), even if this usage makes no sense.

4.2.30 PID_DOWNLOAD_COUNTER (PID = 30)

- Property name: Download Counter
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: DPT_Value_2_Ucount (DPT_ID = 7.010)

4.2.30.1 Abstract Resource definition

The Download Counter shall be a global, Interface Object Type independent read-only Property. A device that has a Download Counter shall at least have PID_DOWNLOAD_COUNTER in the Device Object and may have additional further Interface Objects with PID_DOWNLOAD_COUNTER.

PID_DOWNLOAD_COUNTER in other Interface Objects than the Device Object

Each Download Counter shall be uniquely related to one downloadable part of the device. There may be downloadable parts in the device that are not related to a Download Counter.

PID_DOWNLOAD_COUNTER in the Device Object

If the device has any Download Counter, then it shall at least have PID_DOWNLOAD_COUNTER in the Device Object.

This instance of PID_DOWNLOAD_COUNTER shall signal:

- a change of any further PID_DOWNLOAD_COUNTER in another Interface Object

  If any other PID_DOWNLOAD_COUNTER changes, then also the PID_DOWNLOAD_-COUNTER in the Device Object shall change.

- a manipulation of the memory not protected by another PID_DOWNLOAD_COUNTER.

4.2.30.2 Usage by the Management Server (device)

The Management Server (device) shall autonomously conclude on incrementing the value of the Download Counter when it is unambiguously registered (see below) one or more modifications of one or more bits or octets of the related downloadable part. It shall respond with an incremented value of the Download Counter if the contents of the related downloadable part have changed compared to the preceding read out of the Download Counter.

NOTE 1 Please note that this counter is thus incremented each time, regardless of whether the modification has been done by an authorised person (electrical installer, planner, ETS user, etc.) or not.

The default value of the download counter, this is, the value ex-factory, should be 0000h.

NOTE 2 This is only a recommendation. It is very well possible that ex-factory, the Download Counter differs from 0000h due to
- loading a default application in the device, or
- due to loading a calibration program in the device, or
- due to loading a test application in the device,
etc.

If the Download Counter reaches its maximal value (FFFFh) then it shall not be reset to 0000h!

Unambiguous registration of manipulations (informative)

1. In downloadable parts controlled through a Load State Machine, the transition of the Load State from *Loading* (during which the memory is altered) to *Loaded* unambiguously marks an altered memory contents.
2. In downloadable parts without Load State machine, a simple memory modification through a write-access could be sufficient.

When is the Download Counter incremented and by which value? (informative)

It is not required that the Download Counter increment exactly equals the number of modified bits or octets, or the number of write operations (e.g. A_Memory_Write-PDUs, etc.)

EXAMPLE 1 The Download Counter has an initial value C_1 at the end of a Configuration Procedure and is read out by the Management Client. Afterwards, a first octet is changed in the related downloadable part. This increments the Download Counter to the value C_2 ($C_2 > C_1$). Further modified bits or octets will no longer increment the Download Counter. Now, the Management Client again reads the Download Counter; the device replies with C_2. If now again the related downloadable part is modified, the Download Counter increments to C_3 ($C_3 > C_2$) and will again remain unchanged by further modifications until it is read, by the Management Client.

EXAMPLE 2 One solution for the above can be to work with an odd and an even counter value: the counter is incremented to an even value internally on a change and then does not change anymore with further write accesses. When the counter is read, this even counter will be incremented to an odd counter, sent back, and can from then again be incremented with the first next change.

It is not required that the Download Counter increments immediately if the downloadable part changes. It is only required that a different; incremented value is reported to the Management Client on its next read out of the Download Counter.

EXAMPLE 3 The Download Counter has an initial value C_1 at the end of a Configuration Procedure and is read out by the Management Client. Afterwards one or more bits or octets are changed in the related downloadable part. The Management Server internally notes this situation, but does not yet increment the Download Counter. Only when later the Download Counter is read out, by the A_PropertyValue_Read.ind, the Management Server considers this situation and increments the Download Counter firstly to the value C_2 ($C_2 > C_1$) and then only responds with the A_PropertyValue_Read.res with the value C_2.

What Downloadable Parts shall be governed by the Download Counter?

It is implementation dependent what data is governed by a Download Counter. The application developer concludes and controls what data modification will or will not modify the Download Counter.

A device may have both data that is supervised by a Download Counter as well as data that is not.

4.2.30.3 Usage by the Management Client

The Management Client should firstly read PID_DOWNLOAD_COUNTER in the Device Object. If this PID_DOWNLOAD_COUNTER has not changed, then it may conclude that no further instance of PID_DOWNLOAD_COUNTER in the Management Server has changed value. If PID_DOWNLOAD_-COUNTER in the Device Object has changed, it should discover and read possible further instances of PID_DOWNLOAD_COUNTER in the other Interface Objects to possibly discover the changed memory contents.

The Management Client may read any Download Counter at any time t_1 (DC1) and compare it with any other value of the Download Counter t_2 (DC2) (t_2 can be before or after) and conclude on the possible difference between the read values (DC2 < DC1 or DC2 > DC1) on whether the downloadable part has been changed or not.

4.3 Device Object (Object Type 0)

4.3.1 General requirements

NOTE 6 - In the clauses below, indications are given for each Resource in which existing mask version(s) it is used. This is only to help the reader locating and recognizing the Resource. In future versions, these references will be moved to Volume 9 of the KNX Specifications, where the various BAUs will be specified.

The Device Object shall include information about the device. Properties in the Device Object above 201 shall be manufacturer specific. Except for Easy Parameter blocks and manufacturer specific private parts in E-Mode, application parameters shall be in other Interface Objects than the device Interface Object.

Interface Object: Device Object (object_index 0)				
	Property	PDT (DPT)	Input	Output
	PID_OBJECT_VALUE	PDT_Function	Write: Group Object Handle (2 octets) Value (n octets, n ≤ 10)	ReturnCode (1 octet)
			Read: Group Object Handle (2 octets)	ReturnCode (1 octet) Value (n octets, n ≤ 10)
	PID_OBJECTLINK	PDT_Function	Write: Set/Delete Extended GA (8 octets) Group Object Handle (2 octets)	ReturnCode (1 octet)

			Read: Iterator (1octet)	ReturnCode (1 octet) Iterator (1 octet) Extended GA (8 octets) Group Object Handle (2 octets)
	PID_APPLICATION	PDT_Function	channel number (1 octet) channel code (2 octets)	ErrorCode (1 octet) Or Channel number (1 octet) channel code (2 octets)
	PID_PARAMETER	PDT_Function	Write: channel number (1 octet) parameter number (1 octet) value (n octets, n ≤ 10)	Return Code (1 octet)
			Read: channel number (1 octet) parameter number (1 octet)	Return Code (1 octet) Value (n octets, n ≤ 10)
	PID_OBJECTADDRESS	PDT_Function	Write: Iterator (1 octet)	Return Code (1 octet): always FFh
			Read: Iterator (1 octet)	Return Code (1 octet) Iterator (1 octet) Ch number (1 octet) Ch. Code (2 octets) Grp. Obj. Number (1 octet) Grp. Obj. Handle (2 octets)

4.3.2 PID_ROUTING_COUNT (PID = 51)

- Property name: Routing Count
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall include the default value for the hop count. (This is a parameter for the Network Layer of the device.) With Property based management it shall be used to adjust the default value of the hop count parameter of the Network Layer (BCU normally presents this parameter memory mapped at 10Eh in bit 4-6). Values from 0 to 7 are possible. For details, please refer to the Network Layer parameter specification in [04].

Table 13 - Routing Count

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	reserved	Routing Counter (0…7)			reserved	reserved	reserved	reserved

In Couplers, PID_ROUTING_COUNT shall only be the standard value of the hop count in telegrams of which the Coupler is the initiating transmitter, not the routed telegrams.

4.3.3 PID_MAX_RETRY_COUNT (PID = 52)

- Property name: MaxRetryCount
- Property Datatype: PDT_GENERIC_01
- Datapoint Type: None.

This Property shall include the information of the Data Link Layer parameters nak_retry and busy_retry. This parameter is not adjustable on every device. It depends on the used medium and the implementation. It shall be specified in the Profiles ([15]) whether this parameter shall be adjustable or which fixed value shall be implemented. This Property shall be used in Property based Management to adjust the default value of these parameters of the Data Link Layer (BCU normally presents this parameter memory mapped at 10Fh in bit 2-0 and 7-5).

The default value shall be 3 busy repetitions and 3 Nack repetitions (33h).

Format

Bit	7	6	5	4	3	2	1	0
Meaning	0	busy_retry			0	nak_retry		

4.3.4 PID_ERROR_FLAGS (PID = 53)

- Property name: Error Flags
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

This Property shall contain device specific error flags. Error flag value 0 is set.

Bit	Name	Description	Encoding (d = default value)
0	System 1	Internal System Error EXAMPLE Message system corrupted The message shall be deleted. The system shall be reset. The application shall continue or shall be restarted.	0 = error 1 = ok (d)
1	App	Illegal System State EXAMPLE Invalid value of the Special Function register (SFR) The system shall be reset. The application shall be restarted.	0 = error 1 = ok (d)
2	MEMORY_ERROR	Checksum / CRC-Error in internal non volatile memory EXAMPLE Verify error after flash write operation. The state machine (if present in the device) shall go to the state *Error*. The application shall be stopped.	0 = error 1 = ok (d)
3	Stack	Stack Overflow error The system shall reset. The application shall be restarted.	0 = error 1 = ok (d)
4	Table Error	Inconsistence in System Tables The application shall be stopped.	0 = error 1 = ok (d)
5	Trans	Physical Transceiver error (stuck / overheat detection) The transmission of frames shall be stopped until reset. The application shall continue without the transmission of frames.	0 = error 1 = ok (d)
6	System 2	Internal System error. EXAMPLE Corrupt Memory Table This is only a warning. The application shall continue.	0 = error 1 = ok (d)
7	System 3	Internal System error. This is only a warning. The application shall continue.	0 = error 1 = ok (d)

Please refer to [15] for the specification of which fields (bits) shall be supported by which Profile. Fields that are not supported shall have the default value.

This Property shall be used in Property based Management. This Property corresponds to the error flags of a BCU (located at 10Dh).

4.3.5 PID_PROGMODE (PID = 54)

- Property name: Programming Mode
- Property Datatype: PDT_BITSET8
- Datapoint Type: None.

This Property shall include a binary value that shall be the Programming Mode of the device.

Bit	Name	Description	Encoding (d = default value)
0	PROGMODE	The value of this bit shall reflect the state of the Programming Mode. The value shall be set to 0 on reset. The Management Client can set the device to Programming Mode by setting this bit to 1. This Property shall be used in Property based management and is equal to bit 0 of address 60h of a BCU.	0 = device is not in Programming Mode (d) 1 = device is in Programming Mode
1 to 7	reserved	These bits are reserved and shall always be 0.	not applicable

4.3.6 PID_PRODUCT_ID (PID = 55)

- Property name: Product Identification
- Property Datatype: PDT_GENERIC_10
- Datapoint Type: None.

This Property shall be the manufacturer specific device type. This Property shall enable in LTE-Mode an exact identification of the device (e.g. used by a service tool or for remote operation by a management station).

Name	Offset (Octets)	Size	
Application Manufacturer	0	2 octets	
Device Type	2	8 octets	manufacturer specific device Type

4.3.7 PID_MAX_APDU_LENGTH (PID = 56)

- Property name: MAX. APDU-Length
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

4.3.7.1 Usage by the Management Server

The Management Server shall hold in PID_MAX_APDU_LENGTH the maximal supported APDU-length for Management of the device.

The value of PID_MAX_APDU_LENGTH may be in the range between 15 and 254 [3]).

[3]) 255 as value for PID_MAX_APDU_LENGTH is an ESCape Code (see [02] clause "Length" of the L_Data_Extended-frame).

4.3.7.2 Usage by the Management Client

4.3.7.2.1 End devices

This Property shall allow a Property based Management Client to use the maximal APDU-Length to manage the device. A Management Client supporting the L_Data_Extended-frame has to check this value before starting download.

If the PID_MAX_APDU_LENGTH is not present in the Device Object, then the Management Client shall manage the device with L_Data_Standard-frames with an APDU-length of maximal 15 octets.

4.3.7.2.2 Couplers (Routers)

This shall specify the maximal APDU-length that shall be supported for Management of the Coupler.

If this Property is not present in the Device Object, then the device shall be managed with L_Data_Standard frames.

If PID_MAX_APDU_LENGTH is solely in the Router Object (see 4.4.8), then the device shall only support L_Data_Extended-frames for Routing and only L_Data_Standard-frames for Management.

4.3.7.2.3 cEMI Server

PID_MAX_APDU_LENGTH in the Device Object of a cEMI server

- shall hold the maximum length (of APDU within the frames) that the device shall be able to route from bus to the cEMI client and vice versa, and
- shall be the maximum length (of APDU within the frames) that the device itself shall be able to be managed with (management of the local device).

4.3.8 PID_SUBNET_ADDR (PID = 57)

• Property name:	Subnetwork Address
• Property Datatype:	PDT_UNSIGNED_CHAR
• Datapoint Type:	None.

This Property shall be the Subnetwork Address part (high octet) of the Individual Address.

4.3.8.1 Usage by the Management Server

In a Router, this Property can be used to provide the Subnetwork Address from the Subnetwork attached to its secondary side as specified in “SNA Read” in [09].

4.3.8.2 Usage by the Management Client

This Property can be used by a Management Client (tool or device) for acquiring the Subnetwork Address from the Subnetwork to which it is connected from the Router to that Subnetwork, as specified in “SNA Read” in [09].

4.3.9 PID_DEVICE_ADDR (PID = 58)

• Property name:	Device Address
• Property Datatype:	PDT_UNSIGNED_CHAR
• Datapoint Type:	None.

This Property shall be the Device Address part (low octet) of the Individual Address.

4.3.10 PID_PB_CONFIG (PID = 59)

- Property name: PID_CONFIG_LINK
- Property Datatype: PDT_GENERIC_04
- Datapoint Type: Not applicable.

4.3.10.1 Encoding

The encoding shall always as follows: 4 octets: MSB: Command/Flags; MSB + 1…MSB + 3: data.

Table 14 specifies the coding of the actions that can be performed with PID_CONFIG_LINK.

Table 14 – Overview of actions encoded on PID_CONFIG_LINK

		Value																															
		MSB																										LSB							
		Command				Flags				Data								Data								Data							
Pos.	Action	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
1	Enter_Config_Mode	0	0	0	1	0	0	0	0	00h								00h								00h							
2	Start_Link	0	0	1	0	Flags			Sub function	Manufacturer Code																Number of Group Objects to link							
3	Channel_Function_Actuator	0	0	1	1	0	0	0	0	0	0	0	Channel Code [1]													00h							
4	Channel_Function_Sensor	0	1	0	0	0	0	0	0	0	0	0	Channel Code [1]													00h							
5	Set_Channel_Param	0	1	0	1	Flags				Parameter Index=1								Value								Value							
6	Channel_Param_Response	0	1	1	0	Flags				Parameter Index								Value								Value							
7	Begin_Connection	0	1	1	1	0	0	0	0	00h								00 h								00h							
8	Set_Delete_Link	1	0	0	0	Sub-function				- Connection Code[2] or - scene number value, or - AckSlotNb[3]								Group Address															
9	Link_Response	1	0	0	1	Flags				- Connection Code[2] or - AckSlotNb								Group Address															
10	Stop_Link	1	0	1	0	Flags				00h								00h								00h							
11	Quit_Config_Mode	1	0	1	1	0	0	0	0	00h								00h								00h							
12	Reset_Installation	1	1	0	0	0	0	0	0	00h								00h								00h							
13	Features	1	1	0	1	Sub-function				Physical requirements								reserved								reserved							

[1] E-Mode Channel Code = 13 bit
The E-Mode Channel Code shall be part of the E-Mode Channel information as part of the Device Descriptor Type 2, as specified in clause 4.1.3.1.6 above.

[2] Connection Code = 8 bit
The values Connection Codes are for each E-Mode Channel Specification given in [16]. An overview is given in [13].

[3] AckSlotNb = 6 bit (lsb)

Reserved values shall be set to 0 and shall be checked on reception; in case of discrepancy the message shall be discarded.

Unknown command values (14 and 15) are reserved values. Messages received with these command values shall be discarded and shall not abort the current link-procedure.

4.3.10.2 Action specifications

4.3.10.2.1 Overview

This clause 4.3.10.2 specifies in detail the different actions. Table 15 summarizes all the specified actions.

Table 15 – Actions encoded on PID_CONFIG_LINK - overview

Pos.	Action	Message direction	Service
1	Enter_Config_Mode	Actuator to all	A_NetworkParameter_Write
2	Start_Link	Sensor to all	A_NetworkParameter_Write
3	Channel_Function_Actuator	Actuator to sensor	A_NetworkParameter_Write
4	Channel_Function_Sensor	Sensor to actuator	A_NetworkParameter_Write
5	Set_Channel_Param	Actuator to sensor	A_NetworkParameter_Write
6	Channel_Param_Response	Sensor to actuator	A_NetworkParameter_Write
7	Begin_Connection	Actuator to sensor	A_NetworkParameter_Write
8	Set_Delete_Link	Sensor to actuator	A_NetworkParameter_Write
9	Link_Response	Actuator to sensor	A_NetworkParameter_Write
10	Stop_Link	Sensor to all	A_NetworkParameter_Write
11	Quit_Config_Mode	Actuator to all	A_NetworkParameter_Write
12	Reset_Installation	To all	A_NetworkParameter_Write
13	Features	Sensor to actuator or actuator to sensor	A_NetworkParameter_Write

After the transmission of each message, the expected response shall be received before the 3 s timeout. If the timeout elapses then the procedure shall be aborted (except if the actuator waits for the Features message from the sensor). See Features message for details.

4.3.10.2.2 PID_CONFIG_LINK (Enter_Config_Mode)

		Value										
		Octet 11								Octet 12	Octet 13	Octet 14
		Command				Flags				Data	Data	Data
Pos.	Action	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
1	Enter_Config_Mode	0	0	0	1	0	0	0	0	00h	00h	00h

If the electrical installer wants to link two or more devices, then he at first has to activate Config Mode in a first device. The way how Config Mode has to be activated in any device is implementation specific.

This command shall be sent by a device to be linked when Config Mode becomes active in the device. (The way how Config Mode is activated in a device is implementation specific.)

This action shall activate Config Mode in all sensors and optionally locks all actuators not concerned in this link step.

Open media

This action is optional on open media.

KNX RF Multi

The devices should implement a timeout (same as in KNX RF 1.1) that shall be started when activating the Config Mode. If the Config Mode is not deactivated before the timeout, the devices shall deactivate the Config Mode autonomously. A short timeout is especially useful for the devices that are not in permanent reception mode in runtime. The timeout shall be retriggered each time a configuration message is received in order to enable a long Configuration Procedure.

After the devices have activated the Config Mode, they shall remain in permanent reception mode and the configuration process shall be performed entirely in the F1r RF channels.

4.3.10.2.3 PID_CONFIG_LINK (Start_Link, Flags, Subfunction, Manufacturer Code, Number of Group Objects to link)

		Value																															
		Octet 11								**Octet 12**								**Octet 13**								**Octet 14**							
		Command				**Flags**				**Data**								**Data**								**Data**							
Pos.	**Action**	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
2	Start_Link	0	0	1	0	d Flags	p	Sub function		Manufacturer Code (16 bit)																Number of Group Objects to link							

Flags: d = bit 3: shall indicate to actuators how the link procedure shall be handled.

0 = bidirectional device

1 = unidirectional device

This shall allow systems mixing uni- and bidirectional sensors.

p = bit 2: “parameter indicator” shall indicate whether parameters will be sent in additional messages or not

0 = no additional messages.

1 = messages concerning parameters will be sent after DD2.

Subfunction: 00b: basic mechanism.

Sensors shall send this action after activation (e.g. press push button). It shall indicate to the selected actuator that the link process between this sensor and the actuator that initiated the configuration mode has started. All other sensors are optionally locked after receiving this action.

01b: optional extension.

This action shall be sent by sensors. It shall indicate the activation of Config Mode to link one sensor to several actuators. All others sensors are optionally locked after receiving this action and consider configuration going on (prevents from opening a new session).

Manufacturer Code (16 bit): This shall be sent for information only. No standard action is defined.

Number of Group Objects to link This field shall inform the receiver (actuator) how many Set_Delete_Link actions will follow. If any are lost, for instance due to interferences, then the actuator shall discard any other links received during this process in order to avoid “half-linked” E-Mode Channels.

00h shall mean that the link is valid for all Datapoints common to both devices ,the sender and the receiver. Only one Set_Delete_Link action is necessary.

This is allowed if there is no possible ambiguity in the Datapoints links between these two devices.

Condition: To be secure, this action is usually processed if Config Mode is active. It may however be directly used to start configuration to link one sensor to several actuators. In this case the device needs some specific HMI to initiate the service.

Requirements: Support of Subfunction 00h is mandatory. Others are optional.

4.3.10.2.4 PID_CONFIG_LINK (Channel_Function_Actuator, Channel Code of the actuator)

		Value																															
		Octet 11								Octet 12								Octet 13								Octet 14							
		Command				Flags				Data								Data								Data							
Pos.	Action	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
3	Channel_Function_Actuator	0	0	1	1	0	0	0	0	0	0	0	Channel Code													00h							

After receiving the Start_Link action the actuator shall send the Channel Code of its selected channel to the sensor. If the selected sensor channel is of an appropriate type then the sensor shall select the function according to the received Channel Code.

Expected response: Channel_Function_Sensor

Condition: To be secure, this action shall only be processed in config mode.

4.3.10.2.5 PID_CONFIG_LINK (Channel_Function_Sensor, Channel Code of the sensor)

		Value																															
		Octet 11								Octet 12								Octet 13								Octet 14							
		Command				Flags				Data								Data								Data							
Pos.	Action	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
4	Channel_Function_Sensor	0	1	0	0	0	0	0	0	0	0	Channel Code														00h							

This action shall be the reaction from the sensor after receiving the Channel_Function_Actuator action.

Condition: To be secure, this action shall only be processed in config mode.

4.3.10.2.6 PID_CONFIG_LINK (Set_Channel_Param, Flags, Parameter index, Value)

		Value																															
		Octet 11								Octet 12								Octet 13								Octet 14							
		Command				Flags				Data								Data								Data							
Pos.	Action	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
5	Set_Channel_Param	0	1	0	1	Flags				Parameter Index = 1								Value															

With this action it is possible for actuator to set parameters in sensors.

Flags:

0	0	0	b

0	=	do not override
1	=	override (possibility to change an already set parameter)

Parameter Index: 1 ... 255 (see Channel Code definition)

Value: Parameter value according channel definition (right adjusted)

Expected response: Channel_Param_Response within a delay time [4)]

Condition: To be secure, this action is only processed in config mode

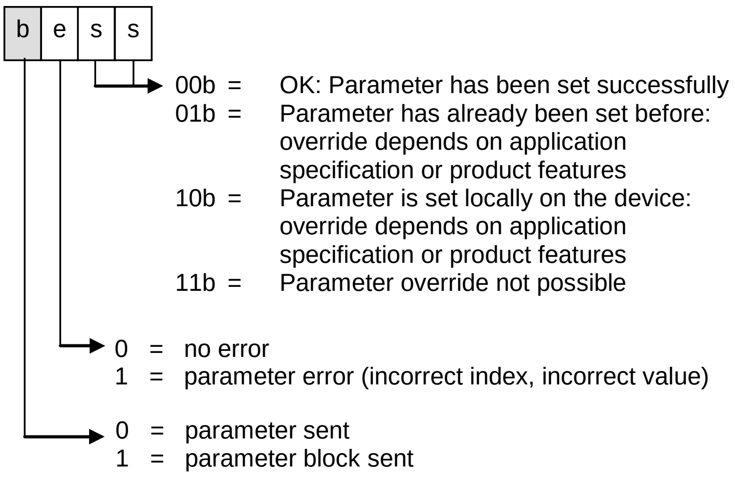
[4)] Depending of medium, a suitable delay time should be selected (typically 1 s for TP1).

4.3.10.2.7 PID_CONFIG_LINK (Channel_Param_Response, Flags, Parameter Index, Value)

		Value																															
		Octet 11								Octet 12								Octet 13								Octet 14							
		Command				Flags				Data								Data								Data							
Pos.	Action	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
6	Channel_Param_Response	0	1	1	0	Flags				Parameter Index								Value															

This action shall be the response from the sensor after receiving the Set_Channel_Param action. It shall include information for the parameter setting.

Flags:



00b	=	OK: Parameter has been set successfully
01b	=	Parameter has already been set before: override depends on application specification or product features
10b	=	Parameter is set locally on the device: override depends on application specification or product features
11b	=	Parameter override not possible

0	=	no error
1	=	parameter error (incorrect index, incorrect value)

0	=	parameter sent
1	=	parameter block sent

- **If b = 0: Parameter is sent**

- Parameter Index:	= 0
	The field *Value* shall be two octets long and contain the channel number and the number of parameters.
	It shall serve to inform the actuator on how many parameters for the current channel number will be sent. If any telegram is lost due to (RF) interferences, the actuator shall discard any other links received during this process in order to avoid "half-linked" channels.
Value:	MSB: channel number
	LSB: number of parameters
- Parameter Index:	N = 1 ... 255 (see Channel Code definition)
Value:	Value of parameter N according channel definition (right adjusted). In case of "parameter locally set" the value shall be the locally set value.
Conditions:	To be secure, this action shall only be processed in config mode.

- **If b = 1: Parameter Block is sent**

- <u>Parameter Index:</u>	= 0: Shall enable a device to send its parameters in one shot after a manufacturer specific interaction, and a receiver to check whether all parameters are received.
	Only parameter blocks with 2 octets can be sent.
Value:	MSB: The value shall be fixed to 0.
	LSB: Shall contain the number of parameter blocks
- <u>Parameter Index:</u>	N = 1 ... 32: gives the channel number in DD2 order declaration.
Value:	Parameter block value of the given channel number. It is sent most significant byte (MSB) and most significant bit (msb) first.
	Unidirectional devices may send the current value of parameters.
	The length of the field Value can be extended to 11 to handle parameters longer than two octets.
Conditions:	None.

EXAMPLE Parameter block of channel CH_Generic_PB_1/2_Info_2 (scene function)

0								8							15
Adjustable selection = 8								▨		Scene Number					
0	0	0	0	1	0	0	0	▨							
msb							lsb								
MSB								LSB							

4.3.10.2.8 PID_CONFIG_LINK (Begin_Connection)

		Value																													
		Octet 11								Octet 12								Octet 13								Octet 14					
		Command				Flags				Data								Data								Data					
Pos.	**Action**	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	Begin_Connection	0	1	1	1	0	0	0	0	00h								00h								00h							

This action shall be sent by the actuator to indicate that the sensor can start the link procedure.

Expected response: Set_Delete_Link within a delay time [5)]

Condition: To be secure, this action shall only be processed in config mode.

[5)] Dependent of the used medium, a suitable delay time should be selected (typically 1 s for TP1).

4.3.10.2.9 PID_CONFIG_LINK (Set_Delete_Link, Sub Function, Connection Code, Group Address)

		Value							
		Octet 11					Octet 12	Octet 13	Octet 14
		Command				Flags	Data	Data	Data
Pos.	Action	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
8	Set_Delete_Link	1	0	0	0	Sub-function			
						0	Connection Code		
						1	scene number value	Group Address	
						2	AckSlotNb		

This PDU shall be sent for each individual Group Object.

Subfunction:

- 0: The sensor shall use this Subfunction to link the Group Objects of the selected E-Mode Channel with the Group Objects of the actuator. Octet 12 shall contain a Connection Code.
- 1: The sensor shall use this Subfunction to associate a scene number Group Object to one GA, indicating at the same time the scene number value to use. It may be repeated for several scene number values. Octet 12 shall contain a scene number value. The Connection Code shall implicitly be the one of a scene number Goup Object.
- 2: This function is designed for the configuration of KNX RF Multi devices.

  The sensor device shall use this Subfunction to associate an ack slot number in KNX to one GA and implicitly to a Group Object.
  - If the Datapoint is an output Datapoint (O Flag), the sensor shall set the value of the ack slot.
  - If the Datapoint is an input Datapoint (I Flag), the value shall have no meaning and the sensor shall wait for the actuator’s Link_Response service to get the ack slot number.

  If no ack is required in runtime for that Extended Group Address then the device shall set the AckSlotNb to the value FFh. The GA field shall not be interpreted in this case and shall be set to 0. The used GA for the RF Multi link shall be taken from the previous Link_Response message corresponding to Set_Delete_Link request (Subfunction 0 or 1).

Requirements: For subfunctions 0 and 1, the RF Multi frequency (Sx or Fx) that is defined for runtime communication shall be deducted from the capabilities of sensor and actuator exchanged with the Features command. If both devices have at least one compatible frequency, this frequency shall be used for runtime (Fx shall be used if the two frequencies are possible). If no frequency is compatible, then the link shall be refused.

Error handling: Discard in case of unknown values.

Expected response: Link_Response within a delay time.

Condition: To be secure, this action shall only be processed while Config Mode is active.

4.3.10.2.10 PID_CONFIG_LINK (Link_Response, Flags, Group Address)

Pos.	Action	Value																															
		Octet 11								Octet 12								Octet 13								Octet 14							
		Command				Flags				Data								Data								Data							
		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
9	Link_Response	1	0	0	1	Flags				Connection Code / AckSlotNb								Group Address															

This action shall be the response from the actuator after receiving the Set_Delete_Link and shall include information about the Link result.

Flags:

3	2	1	0
0	e	s	s

ss:	00b:	Link or AckSlotNb added
	01b:	use existing Group Address or existing AckSlotNb
	10b:	link deleted
	11b:	link or AckSlotNb not added. (The link procedure can be continued.)
e:	0:	No error.
	1:	Error. The link procedure shall be stopped.

Comments: If using the Set_Delete_Link service with subfunction 0 or 1, the flags in the Link_Response service shall apply to the Group Object addition or deletion and shall be relative to the Group Address.

- The required user interaction and HMI for adding or deleting link management is implementation specific.

  EXAMPLE 4 A switch can indicate which operation to perform or a push-button can act as a toggle with indication lamp.

- If another Group Address is already linked to a Group Object, this Group Address shall be used by the sensor device.

  EXAMPLE 5 For a toggle function.

  In this case, the actuator device shall send this action with Flags set to 01b (link already used) and Group Address set with the already linked Group Address).

If using the Set_Delete_Link service with subfunction 2, the flags in the Link_-Response action shall apply to the allocation of the ack slot number to the Group Object having the specified Connection Code. The Group Address field is not used in this case and shall be set to 0. The used Group Address for the RF Multi link shall be taken from the previous Link_Response service corresponding to Set_Delete_Link request (Subfunction 0 or 1).

From the actuator's point of view

- If using the Set_Delete_Link action with subfunction 2 for an input Datapoint (I Flag), the flags in the Link_Response action shall apply to the allocation of the ack slot number by the sensor. The sensor shall set the value of the ack slot number and the actuator shall store the ack slot number for this Group Address in the E-Mode Channel having the specified Connection Code.
- If using the Set_Delete_Link service with subfunction 2 for an output Datapoint (O Flag), the flags in the Link_Response action shall apply to the allocation of the ack slot number by the actuator to the sensor. The actuator shall set the value of the ack slot number and the sensor shall store the ack slot number for this Group Address in the E-Mode Channel having the specified Connection Code.

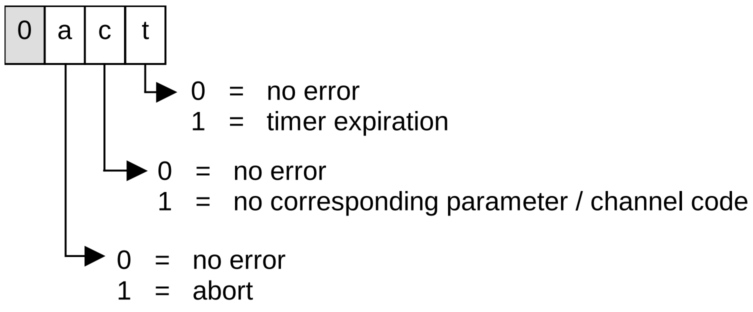
4.3.10.2.11 PID_CONFIG_LINK (Stop_Link)

		Value																															
		Octet 11								Octet 12								Octet 13								Octet 14							
		Command				Flags				Data								Data								Data							
Pos.	**Action**	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
10	Stop_Link	1	0	1	0	Flags				00h								00h								00h							

This action shall be sent by the sensor and shall indicate to the actuator that the link process for the selected channel shall be finished. All sensors shall be unlocked.

In case of abort, all devices shall immediately return to normal mode.

Flags:



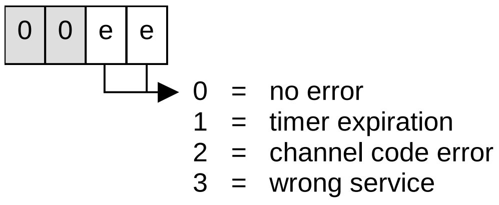
4.3.10.2.12 PID_CONFIG_LINK (Quit_Config_Mode)

		Value																															
		Octet 11								**Octet 12**								**Octet 13**								**Octet 14**							
		Command				**Flags**				**Data**								**Data**								**Data**							
Pos.	**Action**	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
11	Quit_Config_Mode	1	0	1	1	0	0	0	0	00h								00h								00h							

After deactivation of the config mode the actuator shall send the Quit_Config_Mode action.

The sensors shall be set in normal mode after receiving this action and all other actuators shall be unlocked

Flags:



4.3.10.2.13 PID_CONFIG_LINK (Reset Installation)

		Value																															
		Octet 11								**Octet 12**								**Octet 13**								**Octet 14**							
		Command				**Flags**				**Data**								**Data**								**Data**							
Pos.	**Action**	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
12	Reset Installation	1	1	0	0	0	0	0	0	00h								00h								00h							

With this action it shall be possible to reconfigure a device to factory setting (RESET).

Condition: To be secure, this action shall only be processed in config mode.

This action shall not be implemented in KNX-RF devices as no separation from the neighbouring installation is possible.

4.3.10.2.14 PID_CONFIG_LINK (Features, Subfunction, Physical requirements)

		Value							
		Octet 11					Octet 12	Octet 13	Octet 14
		Command				Flags	Data	Data	Data
Pos.	Action	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
13	Features	1	1	0	1	Sub-function			
						0	Physical requirements	reserved	

This action shall be sent in both directions.

Subfunction: - 0: The device shall use this action with the subfunction 0 to send its physical requirements to the other devices. Octet 12 shall contain the physical requirements of the device.

Octet 12 **Physical requirements**

7	6	5	4	3	2	1	0
r	r	r	b4	b3	b2	b1	b0

r: 0: These bits are reserved and shall be 0.

NOTE 7 These may be used for supporting KNX Data Security in the future.

b4: Management (Tx and Rx) of the physical acknowledge (fast Ack)
- 0b: The device shall not be able to manage a physical acknowledge if requested in runtime.
- 1b: The device shall be able to manage a physical acknowledge if requested in runtime.

b3: Transmission on the slow Sx RF channels
- 0b: The device shall not be able to transmit Frames on the slow Sx RF channels.
- 1b: The device shall be able to transmit Frames on the slow Sx RF channels.

b2: Transmission on the fast Fx RF channels.
- 0b: The device shall not be able to transmit Frames on the fast Fx RF channels.
- 1b: The device shall be able to transmit Frames on the fast Fx RF channels.

b1: Scan of the slow Sx RF channels.
- 0b: The device shall not scan the slow Sx RF channels in runtime and thus shall not be able to receive Frames transmitted on a slow Sx RF channel.
- 1b: The device shall not scan the slow Sx RF channels in runtime and thus shall be able to receive Frames transmitted on a slow Sx RF channel.

b0: scan of the fast Fx RF channels

0b: The device shall not scan the fast Fx RF channels in runtime and shall thus not be able to receive Frames transmitted on a fast Fx RF channel.

1b: The device shall scan the fast Fx RF channels in runtime and shall thus be able to receive Frames transmitted on a fast Fx RF channel.

Expected response: After the transmission of the Features (KNX RF Multi Actuator) message, the Features (KNX RF Multi Sensor) message shall be expected within a delay time. If no Features ation is received within this delay, the actuator shall assume that the sensor is a KNX RF Ready sensor. This delay shall have the same value as the delay for all other exchanges of actions, but in this special case, the PB procedure shall continue.

4.3.11 PID_ADDR_REPORT (PID = 60)

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

This Property shall include the KNX Serial Number of the device (6 octets).

If the device has no valid KNX Serial Number then this value shall in datagrams be set to 000000000000h.

4.3.11.1 Usage by the Management Server (device)

The usage by the device shall comply with the Management Server side support of the Management Procedure NM_IndividualAddress_SerialNumber_Report.

4.3.12 PID_ADDR_CHECK (PID = 61)

- Property name: Address Check
- Property Datatype: PDT_GENERIC_01
- Datapoint Type: None.

This Property shall be accessed by a Management Client in the Network Management Procedure NM_IndividualAddress_Check_LocalSubnetwork as specified in [08].

The Management Server (receiving device) shall not interpret the telegram during this procedure, neither give any error message. The telegram shall only be used to receive a L2-acknowledge in that procedure if the tested Individual Address is occupied.

The Property PID_ADDR_CHECK shall not contain data, but only serves for method invocation. As parameter in the procedure NM_IndividualAddress_Check_LocalSubnetwork the value shall consist of a single octet 00h !

4.3.13 PID_OBJECT_VALUE (PID = 62)

- Property name: Object Value
- Property Datatype: PDT_Function
- Datapoint Type: This Property is a Function Property. The coding of the data depends on whether data is written to the function or responded by the function. No single DPT can be given.

4.3.13.1 Abstract Resource Definition

This Property Object Value shall serve to realise the Group Object Indirection as required in [07].

Any access to this Property shall modify the Group Object Flags of the addressed Group Object.

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12	octet...
handle (hi)	handle (lo)	value	value

Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

The function Write ObjectValue shall be used by the client to set the value of a Group Object in the device. Input parameters shall be the Group Object Handle to identify the Group Object in the device and the value to be written to this Group Object.

NOTE 8 The Group Object Handle is specified in [07].

The response shall contain the ReturnCode:

Return Codes:	(00h)	=	SUCCESS (value successfully written)
	(FFh)	=	ERROR (invalid Group Object Handle)

b) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11
handle (hi)	handle (lo)

Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet...
Return Code	value	value	value

The function Read ObjectValue shall be used by the client to read the value of a Group Object in the device. The input parameter is the Group Object Handle to identify the Group Object. The response shall contain the return code and the read value.

Return Codes:	(00h)	=	SUCCESS (value was successfully read)
	(FFh)	=	ERROR (invalid Group Object Handle)

4.3.13.2 Group Object Handle – Realisation Type 1 – RF bidirectional end device

For sending Group Objects this Group Object Handle shall be identical to the pre-assigned sending Extended Group Address.

For receiving Group Objects the Group Object Handle shall be purely regarded as a local object index and shall have the format of a Group Address.

4.3.13.3 Group Object Handle – Realisation Type 2 –RF unidirectional device

For RF unidirectional devices, which only have sending Group Objects, the following rule is defined, how Group Addresses are assigned. The order of Group Objects is defined by

a) the E-Mode Channel definitions and

b) the order in which E-Mode Channels appear in the Device Descriptor Type 2.

In the order defined in this way, Group Addresses shall be assigned to the Group Objects sequentially starting from 0001h.

4.3.13.4 Management Clients

A client (e.g. for visualisation), in order to use the point-to-point access, must know the values of the Group Addresses and Group Object Handles of the Group Objects present in the device.

For an E-Mode Management Server, the client can derive the active Group Objects in the device from the Device Descriptor Type 2 (DD2) in combination with the knowledge contained in the Channel database.

A central Management Client shall have knowledge of the object handle values from its product database.

However, in the general case, it is not possible to know all Group Objects in the device, because some could be hidden, this is, do not appear in DD2 in the current hardware configuration. This is the case e.g. if a generic communication module can be attached to several actuators.

For such Profiles, PID_OBJECT_ADDRESS is required. This Property shall enable reading of the Group Addresses and object handles by the Management Client.

4.3.14 PID_OBJECTLINK (PID = 63)

• Property name:	Object Link
• Property Datatype:	PDT_Function
• Datapoint Type:	None.

Used by: mask 2010h
RF bidirectional devices

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21
Set/Delete	00h	SN (high)	SN	SN	SN	SN	SN (low)	GA (high)	GA (low)	handle (hi)	handle (lo)

Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

The function Write ObjectLink shall be used by a Management Client to assign a new receiving Group Address to a Group Object or to delete a Group Address from a Group Object. Input parameters shall be the Group Object Handle to identify the Group Object, and the Extended Group Address to be assigned/deleted.

Note that the Extended Group Address may include the own Serial Number of the device, in this case an internal assignment shall be established / deleted.

Octet 10 (set/delete) decides whether a link is set or deleted:

Codes:	(00h)	= set link
	(01h)	= delete link

The response shall contain the return code of the operation.

Return Codes:	(00h)	= SUCCESS (assignment /deletion done or already existed / net present)
	(FFh)	= ERR_TABLEFULL (no more space in link table)
	(FEh)	= ERR_OBJECT (Group Object does not exist)

b) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11
00h	Iterator

Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21
Return Code	Iterator	SN (high)	SN	SN	SN	SN	SN (low)	GA (high)	GA (low)	handle (hi)	handle (lo)

The function Read ObjectLink shall be used by a Management Client to read all Extended Group Addresses assigned to the Group Objects in a device. Therefore this Property serves as an iterator over the Group Object Association Table. The input parameter is an iterator that counts the links in the device.

The response shall repeat the iterator value from the Read and contains the return code 00h, the Extended Group Address and the associated Group Object Handle in case an additional Extended Group Address is available. If no more link is present, the return code is FFh.

4.3.15 PID_APPLICATION (PID = 64)

- Property name: Application
- Property Datatype: PDT_Function
- Datapoint Type: None.

Used by: This Property is not mandatory for any of the existing KNX Profiles.

a) Write (A_FunctionPropertyCommand_PDU)

octet 10	octet 11	octet 12	octet 13
00h	E-Mode Channel Number	Connection Code	

Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13
Return Code	E-Mode Channel Number	Connection Code	

The function Write Application shall be used by a Management Client to change the running application in a device, provided that the device supports multiple applications.

EXAMPLE A push button may support the applications “switching”, “dimming” and “blinds”.

Input parameters shall be the Channel Number of the E-Mode Channel of which the application is to be changed, and the E-Mode Channel Code of the new application. The response shall repeat the E-Mode Channel Number and the new channel code. In case the switch to the desired channel code is not possible, the previously used channel code shall be returned.

Return Codes:	(00h)	=	SUCCESS (application changed)
	(FFh)	=	ERROR (invalid channel number or channel code)

b) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11
00h	E-Mode Channel Number

Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13
Return Code	E-Mode Channel Number	Connection Code	

The function Read Application shall be used by a Management Client to check which application is running in a device.

Return Codes:	(00h)	=	SUCCESS (channel code successfully read)
	(FFh)	=	ERROR (invalid channel number)

4.3.16 PID_PARAMETER (PID = 65)

- Property name: Parameter
- Property Datatype: PDT_Function
- Datapoint Type: None

Used by: mask 2010h
RF bidirectional devices

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12	octet …
E-Mode Channel Number	Parameter Number	Value	Value

Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

The function Write Parameter shall be used by a Management Client to change an application parameter in a device. Input parameters shall be the E-Mode Channel Number of the E-Mode Channel to which the parameter belongs, the parameter number (according to E-Mode Channel definitions), and the parameter value. The output parameter shall be a return code.

Return Codes:	(00h)	=	SUCCESS (value successfully written)
	(FFh)	=	ERROR (invalid parameter)

b) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11
E-Mode Channel Number	Parameter Number

Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet …
Return Code	00h	Value	Value

The function Read Parameters shall be used by a Management Client to read an application parameter from a device. Input parameters shall be the channel number of the channel to which the parameter belongs, and the parameter number (according to channel definitions). Output parameter shall be a return code and the value read.

Return Codes:	(00h)	=	SUCCESS (value successfully written)
	(FFh)	=	ERROR (invalid parameter)

Return Codes: (00h) = SUCCESS (value successfully written)

(FFh) = ERROR (invalid parameter)
In this case, the A_FunctionPropertyState_Response-PDU shall end at octet 10 “Return Code”. There shall be no octet 11 and no value (no octet 12 or beyond).

4.3.17 PID_OBJECTADDRESS (PID = 66)

- Property name: Object Address
- Property Datatype: PDT_Function
- Datapoint Type: None.

Used by: mask 2010h
RF bidirectional devices

a) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11
00h	Iterator

Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17
Return Code	Iterator	E-Mode Channel Number	E-Mode Channel Code		Group Object Number	Group Object Handle	

The function Read ObjectAddress shall be used by a Management Client to read the Group Object Handle of the objects from a device. Every response shall contain the Group Object Handle of one Group Object.

The Read shall contain an Iterator that shall count the Group Objects in the device. The Management Client shall issue Reads with the Iterator starting from 0 in a continuous order.

The Response shall contain the same Iterator value in case of a valid Group Object Handle. In this case the E-Mode Channel Number (corresponding to the order in DD2), the E-Mode Channel Code to which the Group Object belongs, the Group Object number in the E-Mode Channel (according to the E-Mode Channel definitions, see there), and the Group Object Handle shall follow. The Return Code shall in this case be 00h.

In case there is no corresponding Group Object value present in the device for an Iterator value, the Return Code shall be FFh. The E-Mode Channel Number and Group Object number shall be counted from 0.

b) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11
00h	Iterator

Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

The function Write ObjectAddress has no functionality. The Return Code shall in any case be FFh.

4.3.18 PID_PSU_TYPE (PID = 67)

• Property name:	PSU Type
• Property Datatype:	PDT_UNSIGNED_INT
• Datapoint Type:	DPT_UElCurrentmA (7.012)
• Memory type:	Non-volatile memory.

4.3.18.1 Abstract Resource definition

This Property shall contain the nominal supply current [mA] of the bus power supply in the device.

4.3.18.2 Format

This supply current value shall be binary encoded using a 16 bit unsigned integer with 1 mA resolution. The value 0 is reserved (no bus power supply functionality available).

4.3.18.3 Usage by the Management Client

This Property can be used by a Management Client to scan presence of PSUs/DPSUs in a system in order to calculate the total available supply current or to activate/deactivate individual PSU/DPSU.

To this, the Management Client shall use:

- the General Procedure for accessing Properties with A_NetworkParameter_Read

```
    /* Scan for the presence of PSU and DPSU in the network */
    /* hop_count_type = 0 only scans the Subnetwork in which the Management Client is located. */
    /* hop_count_type = standard scans the entire installation. */
NM_NetworkParameter_Read_R(hop_count_type, parameter_type = [object_type =Device Object,
    Property_id = PID_PSU_TYPE], priority = system, test_info = 00h)
    /* Every DPSU or PSU with communication capabilities now reports individually */
    /* the nominal current of its PSU or DPSU. */
```

- DMP_InterfaceObject_Read_R with A_PropertyValue_Read

Both procedures are specified in [08].

4.3.18.4 Usage by the Management Server (device)

This Property shall be read only.

This Property shall be readable through A_NetworkParameter_Read and A_PropertyValue_Read.

The implementation of this Property is mandatory if the bus power supply feature is implemented in a communicating device supporting Property based management. For other realisations, as e.g. stand alone bus power supply units, the implementation of this Property is conditional on the DPSU Profile in [15].

4.3.19 PID_PSU_STATUS (PID = 68)

- Property name: PSU Status
- Property Datatype: PDT_BINARY_Information
- Datapoint Type: DPT_Switch (1.001)
- Memory type: Volatile memory.

4.3.19.1 Abstract Resource definition

This Property shall indicate whether the bus power supply in the device is currently switched on or off. This information is mainly used for diagnostics or visualisation.

4.3.19.2 Format

The Property Value shall be encoded according DPT_Switch (1.001) as specified in [12].

NOTE 9 “0”: “the power supply in the device is switched off”;
“1”: “the power supply in the device is switched on”

4.3.19.3 Usage by the Management Client

This Property can be used by a Management Client to retrieve the status of all PSUs and DPSUs in a system with communication capabilities to check the state of the bus power supply feeding functionality.

To this, the Management Client shall use:

- the General Procedure for accessing Properties with A_NetworkParameter_Read

```
    /* Scan for the status of PSUs and DPSUs */
    /* hop_count_type = 0 only scans the Subnetwork in which the Management Client is located. */
    /* hop_count_type = standard scans the entire installation. */
NM_NetworkParameter_Read_R(hop_count_type, parameter_type = [object_type =Device Object,
    Property_id = PID_PSU_STATUS], priority = system, test_info = 00h)
    /* Every DPSU or PSU with communication capabilities now reports individually */
    /* the state of its PSU or DPSU. */
```

- DMP_InterfaceObject_Read_R with A_PropertyValue_Read, or

Both Management Procedures are specified in [08].

4.3.19.4 Usage by the Management Server (device)

This Property shall be read only.

The implementation of this Property is mandatory if the bus power supply feature is implemented in a communicating device supporting Property based management. For other realisations, as e.g. stand alone bus power supply units, the implementation of this Property is conditional on the DPSU profile in [15].

4.3.20 PID_PSU_ENABLE (PID = 69)

- Property name: PSU Enable
- Property Datatype: PDT_ENUM8
- Datapoint Type: DPT_PSUMode (20.008)
- Memory type: Non-volatile memory.

4.3.20.1 Abstract Resource definition

This Property shall be used to control operation of the bus power supply from a Management Client.

4.3.20.2 Format

For the encoding of the Property Value, please refer to DPT_PSUMode (20.008) in [12]. The supported range is product specific; e.g. disabled/enabled only or disabled/auto only.

4.3.20.3 Usage by the Management Client

Access to this Property using A_NetworkParameter_Read and A_NetworkParameter_Write services is not allowed. Broadcast write of PID_PSU_ENABLE could activate more than the allowed 8 DPSU and/or 2 central PSU or could deactivate all PSU/DPSU and communication would break down.

4.3.20.4 Usage by the Management Server (device)

Implementation of this Property is optional. PSU/DPSU can be activated/deactivated also by mechanical means (e.g. a jumper or a switch).

4.3.21 PID_DOMAIN_ADDRESS (PID = 70)

• Property name:	Domain Address
• Property Datatype:	PDT_UNSIGNED_INT
• Datapoint Type:	DPT_Value_2_Ucount

This Property shall include the Domain Address of the Device

The Domain Address shall be managed through the appropriate Management Procedures as specified in [08].

EXAMPLES NM_DomainAndIndividualAddress_Read, NM_DomainAndIndividualAddress_Write

The Property based access to the Domain Address is interesting for local access, e.g. through cEMI. The value of the Property shall be the Domain Address of the cEMI server device itself, this is if the cEMI server device should be in only one Domain as a Management Server, seen from the bus medium.

4.3.22 PID_IO_LIST (PID = 71)

• Property name:	Interface Object List
• Property Datatype:	PDT_UNSIGNED_INT
• Datapoint Type:	None

PID_IO_LIST shall be a read-only, array structured Property with as many elements as Interface Objects are located (or “active”) in the device. The array element with index 0 shall contain the number of array elements (or Interface Objects) implemented in the Device. The Property shall contain the Interface Object Types in ascending order, starting with the array field with index 1. For Interface Objects with more than 1 instance, the 1st instance shall be at the position with the lower/lowest index; the Interface Object Type of the 2nd instance shall be placed at the next index and so on.

4.3.23 PID_MGT_DESCRIPTOR_01 (PID = 72)

• Property name:	Management Descriptor 1
• Property Datatype:	PDT_GENERIC_10
• Datapoint Type:	None

4.3.23.1 Use

This Property shall be read-only and shall provide an identification for the management of the device.

4.3.23.2 Format

The Property Value shall be a 10 octet value, of which octet 1 (MSB) shall contain the field "Structure indication" and shall specify the structure of the further contents of this Management Descriptor 1. The coding and possible values of octets 2 to 10 shall depend on the value given in the field "Structure indication" in octet 1.

Octet	1 (MSB)	2	3	4	5	6	7	8	9	10 (LSB)
Field	Structure indication	structure contents								

Field "Structure indication" (octet 1)

The datatype of the field "Structure indication" shall be enumeration (datatype N_8).

Value	Management Model	Remark
0	reserved	
1	Property-based implementation independent management on TP1	System 300, mask 0300h
2	Property-based implementation independent management on RF for BiBat devices	System 300, mask 2300h
3 to 255	reserved	Reserved for future extensions

4.3.23.3 Structure indication 01h: devices supporting System 300

Octet	1 (MSB)	2	3	4	5	6	7	8	9	10 (LSB)
Field	Structure indication	Reserved		Table Manage ment	Reserved					
Value	01h	00h	00h		00h	00h	00h	00h	00h	00h

For devices supporting System 300 on TP1 the field "Structure indication" (octet 1) shall have the value 01h.

Octets 2 and 3 shall be reserved and shall have the value 00h.

The field "Table Management" (octet 4) shall contain indications to the management models to be used by a Management Client for Group Address Table management and Group Object Association Table management.

Octets 5 to 10 shall be reserved and shall have the value 00h.

Field “Table Management” (octet 4)

This field shall be divided in two subfields of each one nibble.

Bit	7 (MSB)	6	5	4	3	2	1	0
Field	Group Address Table Management				Association Table Management			
Value	0h				0h or 1h			

The datatype of each of the two fields shall be an enumeration (Datatype N_4).

The higher nibble is foreseen to distinguish between different Group Address Table management models. Since currently no different such management models are needed for System 300, this nibble is still reserved and shall have the value 0h.

The lower nibble shall be an indication to the Group Object Association Table management model to be used by a Management Client.

The PDT indicated by this field shall be according the format of the Property based Group Object Association Table as specified in PID_TABLE (PID = 23).

Value	PDT for PID_TABLE
0	PDT_GENERIC_02[ ] 6)
1	PDT_GENERIC_02[ ] or PDT_GENERIC_04[ ], to be discovered from the Property description of PID_TABLE 7)
2 to 15	reserved values

4.3.23.4 Structure indication 02h: RF BiBat devices supporting Property-based management

For BiBat devices the field “Structure Indication” (octet 1) shall have the value 02h.

The field “Management Model’ (octet 2) shall contain the supported basic Management Model.

The field “BiBat Feature” (octet 3) shall contain the supported BiBat features.

The field “Table Management” (octet 4) shall contain indications to the Management Models to be used by a Management Client for Group Address Table management and Group Object Association Table management.

Octets 4 to 10 shall be reserved and shall have the value 00h.

Octet	1	2	3	4	5	6	7	8	9	10
Field	Structure Indication	Management Model	BiBat Feature	Table Management	Reserved					
Value	02h				00h	00h	00h	00h	00h	00h

6) ETS: application programs created for use with the system 300 system plug-in.

7) ETS: application programs created for use with the native ETS-built-in support for system 300 devices.

Field “Management Model” (octet 2)

The datatype of this field shall be an enumeration (Datatype N_8).

value	Management Model	Remark
0	reserved	
1	Property-based implementation independent management	System 300, LTE Device
2	RF S-Mode	S-Mode Management for bidirectional RF device according [09], clause 2.4 “RF bidirectional devices”
3 to 255	reserved	

Field “BiBat Feature” (octet 3)

The datatype of this field shall be a bitset (Datatype B_8).

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
reserved 0b	reserved 0b	reserved 0b	BiBat FastACK	BiBat LongHeader	BiBat Retransmitter	BiBat BiBat Slave	BiBat Master

Combined BiBat features are possible. Table 16 shows the allowed combinations. Other combinations are not allowed.

Table 16 – Allowed values of the field BiBat Feature

Field name						
reserved b_7 to b_5	FastACK b_4	LongHeader b_3	Retransm b_2	BiBat Slave b_1	BiBat Master b_0	Description
000b	0	0	0	0	1	BiBat Master, no support of fast ACK
000b	1	0	0	0	1	BiBat Master, support of fast ACK
000b	0	0	0	1	0	pure BiBat Slave
000b	1	0	0	1	0	BiBat Slave expecting feedback to own action
000b	0	1	0	0	0	BiBat Alarm device, long header, asynchronous only
000b	0	1	0	1	0	BiBat Alarm device, long header, supporting synchronous comm. as BiBat Slave
000b	0	0	1	0	0	BiBat Retransmitter

Field “Table Management” (octet 4)

The encoding shall be identical as the encoding for Structure indication 01h.

4.3.16.4 Other formats/structures

Other formats shall not be used and are reserved by KNX Association for future extensions.

4.3.24 PID_PL110_PARAM (PID = 73)

- Property name: PL110 Parameters
- Property Datatype: PDT_GENERIC_01
- Datapoint Type: None.

This is a PL110 specific parameter for network management.

Table 17 - Base Configuration for PL110

Bit	7	6	5	4	3	2	1	0
Field name	reserved	Frequency definition		Repeater installed	reserved	Repeater mode	reserved	reserved
Coding or Value	1	11b: channel A 10b: reserved 01b: reserved 00b: channel B		0: enabled 1: disabled	1	0: device is Repeater 1: device is no Repeater	1	1

For frequency definition 'channel' means a set of two frequencies for the FSK (frequency shift keying) transmission of the PL110 devices.

When the Management Client changes the frequency definition of a device it will immediately loose connection when the new channel is set.

For the setting of the repeater flags the following table has to be applied:

Table 18 - Setting of repeater flags

Function	Flag "Repeater installed"	Flag "Repeater Mode"
• no repeater present in the installation • device itself is not repeater	1	1
• not allowed	1	0
• repeater present in the installation • device itself is not repeater	0	1
• repeater present in the installation • device itself is repeater	0	0

4.3.25 PID_RECEIVE_BLOCK_TABLE (PID = 75)

- Property name: BiBat Receive Block Table
- Property Datatype: PDT_UNSIGNED_CHAR[16]
- Datapoint Type: None.

The Property BiBat Receive Block Table shall contain the activated receive blocks in the BiBat Slave, which are assigned by the BiBat Master. The BiBat Slave shall activate its receiver in the corresponding receive blocks.

The BiBat Master shall assign receive blocks to each BiBat Slave via a 128 bit table at configuration time and may alter this receive block bit table as long as the average number of receive blocks signalled by the BiBat Slave is not exceeded. The BiBat Master shall assign each such additional receive block to a single BiBat Slave, a group of BiBat Slaves or to all BiBat Slaves.

Any BiBat Slave shall be able to receive sync or data frames in the first time slot of block #0 and block #64 of each section, independent of the value of the Property BiBat Receive Block Table.

In the standard system the data length of a Property element is restricted to 10 octets. Therefore the table shall be encoded as an array Property with 16 elements (16 x 1 octet). The BiBat Master shall use two or more messages to write the complete Property BiBat Receive Block Table. Because of the segmented write access to the table, the BiBat Slave cannot check, whether the table data is consistent. The BiBat Master shall ensure complete download of the table.

Table 19 – Receive block Table

Data format

Element 16	Element 15	Element 14	Element 13	Element 12	Element 11	Element 10	Element 9
Block # 127-120	Block # 119-112	Block # 111-104	Block # 103-96	Block # 95-88	Block # 87-80	Block # 79-72	Block # 71-64

Element 8	Element 7	Element 6	Element 5	Element 4	Element 3	Element 2	Octet 1
Block # 63-56	Block # 55-48	Block # 47-40	Block # 39-32	Block # 31-24	Block # 23-16	Block # 15-8	Block # 7-0

Element 16							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Block # 127	Block # 126	Block # 125	Block # 124	Block #123	Block #122	Block #121	Block #120
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
……..							
Element 1							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Block # 7	Block # 6	Block # 5	Block # 4	Block #3	Block #2	Block #1	Block #0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	1

Each of the 128 bits of the table shall represent a block. The mapping of the block# to the corresponding bit in the table is shown above.

If the bit is set to ’1’, the corresponding receive-block shall be active.

If the bit is set to ’0’, the corresponding receive-block shall be inactive.

The table fields for block #0 and blocks #64 shall always contain the value ‘1’.

The table shall be stored by the BiBat Slave in non-volatile memory.

4.3.26 PID_RANDOM_PAUSE_TABLE (PID = 76)

- Property name: BiBat Random Pause Table
- Property Datatype: PDT_UNSIGNED_CHAR[12]
- Datapoint Type: None.

Every BiBat Master shall maintain a unique table of 12 values for the length of 12 subsequent random pauses. It shall distribute this random pause table to all its BiBat Slaves during configuration. The table shall be derived from the unique 48 bit KNX Serial Number of the BiBat Master.

Each table element shall be encoded on 8 bit (transport format).

- The allowed range of each value is 0 to 15 (lower 4 bit) corresponding to a random pause length of 0 s to 0,937 5 s.
- The upper 4 bit of each table element are fixed 0000b.

In the BiBat Slave the table may be stored in a packed format (e.g. two 4 bit table values in one octet).

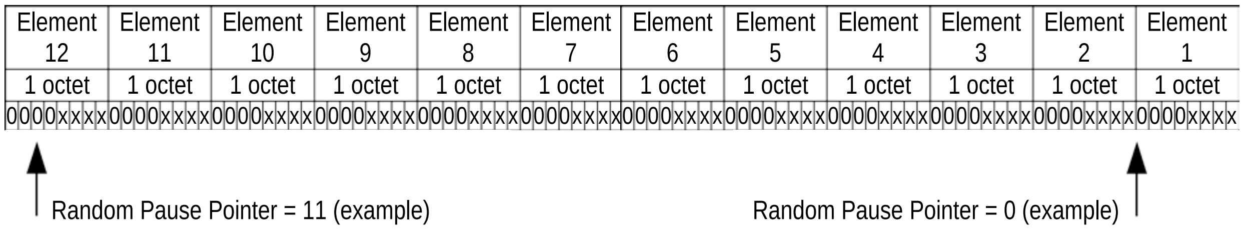
In the standard system the data length of a Property element is restricted to 10 octets. Therefore the table shall be encoded as an array Property with 12 elements (12 x 1 octet). The BiBat Master shall use two or more messages to write the complete Property BiBat Random Pause Table. Because of the segmented write access to the table, the BiBat Slave cannot check, whether the table data is consistent. The BiBat Master shall ensure complete download of the table.

The BiBat Slave shall store the table in non-volatile memory

To allow the BiBat Slave to resynchronise fully to the BiBat Master, the 'Help Call Response' message shall contain a random pause pointer value (0 to 11).

- Random pause pointer value 0 to 11: points to the corresponding element 1 to 12 of the random pause table (n+1)
- 13th random pause, fixed length 1 s (value 10h)

Datapoint format



Element 12	Element 11	Element 10	Element 9	Element 8	Element 7	Element 6	Element 5	Element 4	Element 3	Element 2	Element 1
1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet
0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx	0000xxxx

Random Pause Pointer = 11 (example)
Random Pause Pointer = 0 (example)

4.3.27 PID_RECEIVE_BLOCK_NR (PID = 77)

- Property name: BiBat Receive Block Number
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

A synchronous BiBat Slave may signal a higher receive frequency (= smaller reception period) to the BiBat Master at configuration time. The Property BiBat Receive Block Number shall contain the (average) number of additional receive blocks (0 to 126) within a 128 block section in addition to the minimum (default). The BiBat Master shall check the number of additional receive blocks during configuration and shall then assign receive blocks to each BiBat Slave via the 128 bit Property BiBat Receive Block Number. The BiBat Master may alter this Property BiBat Receive Block Number as long as the average number of receive blocks signalled by the BiBat Slave is not exceeded.

The Property is mandatory and shall be read-only.

Value 0 is allowed and shall mean "no additional receive-blocks".

4.3.28 PID_HARDWARE_TYPE (PID = 78)

- Property name: Hardware Type
- Property Datatype: PDT_GENERIC_06
- Datapoint Type: None.

The Property PID_HARDWARE_TYPE in the Device Object shall identify the hardware.

PID_HARDWARE_TYPE shall contain a single value, encoded as specified below, and shall not use the array aspect of Property value (single array element Property). One hardware shall thus have only one value for PID_HARDWARE_TYPE.

Coding of PID_HARDWARE_TYPE

The high octet of this Property shall specify the format of the other five octets:

High octet	Meaning of following 5 octets	Usage
00h	Manufacturer specific device identification.	See Volume 6 ([15]) for the mandatory and optional use of PID_HARDWARE_TYPE..
Other values	To be defined	Reserved for future standard use.

This Property shall be an identifier for the device hardware. It shall specify the hardware of a device in the sense of a product group, i.e. the Management Client shall allow the download of one or more applications into the device with a specific value for Hardware Type.

The MSB shall be 00h. Other values for the MSB shall not be used. The definition of the MSB is under the supervision of the KNX Association System Group.

4.3.29 PID_RETRANSMITTER_NUMBER (PID = 79)

- Property name: BiBat Retransmitter Number
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: None.

The Property BiBat Retransmitter Number shall be assigned by the BiBat Master during configuration and shall define time delay of the Retransmitter.

Range: values 1 to 3 are allowed values that shall be assigned by the BiBat Master.

The default value in the Retransmitter at manufacturing time shall be 0. The value 0 shall indicate that the Retransmitter is unconfigured, this is, it is not active for synchronous communication

The Retransmitter shall store the value of the Property BiBat Retransmitter Number in non-volatile memory.

4.3.30 PID_SERIAL_NR_TABLE (PID = 80)

- Property name: KNX Serial Number Table
- Property Datatype: PDT_GENERIC_06[]
- Datapoint Type: DPT_SerNum (221.001)

4.3.30.1 Use

The KNX Serial Number Table Property shall contain the KNX Serial Numbers of the associated RF transmit-only devices.

Implementation of the KNX Serial Number Table is mandatory in RF bidirectional devices if the product has the ability to interwork with transmit-only devices.

Reading out of the KNX Serial Number Table by a Management Client, tool, like ETS, shall be possible. Setting of the KNX Serial Number Table by a Management Client, like ETS, is optional.

4.3.30.2 Format

The KNX Serial Number Table Property shall be an array of 6 octet KNX Serial Numbers. The table shall contain the KNX Serial Numbers of linked devices or void table elements as specified below.

Array Index	Value
1	1st KNX Serial Number (6 Octets)
2	2nd KNX Serial Number (6 Octets)
3	….
…	…
N	…

Figure 3 – Structure of the KNX Serial Number Table

4.3.30.3 Usage by the Management Server (device)

4.3.30.3.1 Usage by LTE RF BD devices

The KNX Serial Number Table shall be generated by the device itself during linking with RF transmit-only devices.

– For each additional RF transmit-only partner device that is linked to this Management Server (device) the corresponding KNX Serial Number shall be added in the list (last element + 1). The table elements need not to be sorted.

– The maximal size of the table shall be fixed and product specific and the number of devices that can be associated is therefore limited.

– Ex-factory, the table shall be empty and void table elements shall contain the KNX Serial Number 000000000000h.

– There may be gaps in the table with void table elements.

– Handling of full table

  In case of insufficient table size the handling can be manufacturer specific; it is recommended to generate an error indication and not to overwrite existing table elements.

– Deletion of individual table elements

  The deletion of individual links is not foreseen in this standard but it shall be possible to reset the whole table by manufacturer specific means.

– Write access to the table shall be optional (e.g. by ETS)

– Individual table elements can be written. There is no table load state machine.

– The current length N of the table Property shall be readable by means of an A_PropertyDescription_Read-service.

4.3.30.3.2 Usage by BiBat Retransmitter

The Property in the BiBat Retransmitter shall at least contain the KNX Serial Number of the BiBat Master of which the BiBat Retransmitter shall retransmit the synchronous and asynchronous telegrams.

The KNX Serial Number of the BiBat Master shall be assigned to element 1 of the table.

NOTE 10 BiBat Retransmitters shall retransmit synchronous telegrams only with the Domain Address of the BiBat System or the KNX Serial Number of the BiBat Master. Selective retransmission/filtering (according to KNX Serial Numbers) of asynchronous telegrams from other devices is an optional feature of the BiBat Retransmitter.

In addition to the BiBat Master KNX Serial Number, the table may contain the KNX Serial Numbers of other devices for selective retransmission of asynchronous telegrams.

Rule for selective retransmission and filtering

a) If the Property KNX Serial Number Table contains only one element (i.e. the KNX Serial Number of the BiBat Master) then

- asynchronous telegrams containing any KNX Serial Number shall systematically be retransmitted; and
- asynchronous telegrams containing the Domain Address of the BiBat System shall be retransmitted; and
- synchronous telegrams with the corresponding KNX Serial Number of the BiBat Master shall be retransmitted, all other synchronous telegrams shall be discarded.

b) If the Property KNX Serial Number table contains more than one element then

- asynchronous telegrams containing the KNX Serial Number of the sender shall be filtered according to the Property KNX Serial Number Table; and
- asynchronous telegrams containing the Domain Address of the BiBat System shall be retransmitted; and
- synchronous telegrams with the KNX Serial Number of the BiBat Master shall be retransmitted, all other synchronous telegrams shall be discarded.

The Property KNX Serial Number table shall be set by the BiBat Master at configuration-time and shall be stored in non-volatile memory.

The Property KNX Serial Number Table shall be an array of 6 octet KNX Serial Numbers. The table shall contain KNX Serial Numbers used by the retransmitter for filtering or void table elements as specified below.

- The size of the table shall be fixed and product specific and the number of devices that can be associated shall therefore be limited.
- The length of the table Property shall be readable by means of A_PropertyDescription_Read.
- The BiBat Master shall check the maximum size of the table using A_PropertyDescription_Read before writing the table.
- The table shall have no load state machine. Therefore all elements shall be written by the BiBat Master.
- The table elements need not to be sorted. Element 1 shall however contain the KNX Serial Number of the BiBat Master.
- Ex-factory, the table shall be empty and void table elements shall contain the KNX Serial Number 000000000000h.
- There may be “gaps” in the table with void table elements.
  - Erasing individual table elements is possible by writing the KNX Serial Number 000000000000h to the corresponding element.
  - Write access to the table is mandatory in the BiBat retransmitter.

Array Index	Value
1	1st KNX Serial Number (6 octets): of the BiBat Master
2	2nd KNX Serial Number (6 octets): of any other device
3	3rd KNX Serial Number (6 octets): of any other device
…	…
N	…

Figure 4 – Encoding of the KNX Serial Number Table

4.3.31 PID_BIBAT_MASTER_ADDRESS (PID = 81)

• Property name:	BiBat Master Individual Address
• Property Datatype:	PDT_UNSIGNED_INT
• Datapoint Type:	None.

The Property BiBat Master Individual Address shall contain the Individual Address of the BiBat Master. The BiBat Slave needs the BiBat Master's Individual Address as Destination Address in the help call.

This Property shall be set by the BiBat Master at configuration-time.

The BiBat Slave shall store the value in non-volatile memory.

4.3.32 PID_RF_DOMAIN_ADDRESS (PID = 82)

• Property name:	RF Domain Address
• Property Datatype:	PDT_GENERIC_06
• Datapoint Type:	None.

4.3.32.1.1 PID_RF_DOMAIN_ADDRESS in the Device Object of the cEMI Server

The Property PID_RF_DOMAIN_ADDRESS shall contain the value of the Domain Address of the cEMI Server that shall be used for filtering or that shall be inserted in the transmitted RF frame if the L_Data.req contains the Domain Address 000000000000h; see [11] clause 4.1.4.3.2 "RF medium information".

NOTE 11 This is a different Property than PID_DOMAIN_ADDRESS (PID = 70) that is used on the Powerline communication medium and that has a size of only two octets.

4.3.33 PID_DEVICE_DESCRIPTOR (PID = 83)

• Property name:	Device Descriptor
• Property Datatype:	PDT_GENERIC_02
• Datapoint Type:	None.

This Property shall contain the Device Descriptor Type 0 of the device.

The format and coding shall be identical to what is specified in clause 4.1.2.

The value of this Property shall be identical to the value of the device's Device Descriptor Type 0 value as accessed though any other method, such as (not exclusive)

- the Device Descriptor Type 0 value provided using the dedicated Application Layer service A_DeviceDescriptor_Read. For use please refer to [08], e.g. Management Procedures NM_IndividualAddress_Write, NM_IndividualAddress_SerialNumber_Write2, NM_DomainAndIndividualAddress_Write, NM_DomainAndIndividualAddress_Write2, NM_IndividualAddress_Check, DMP_Connect_RCo, DMP_Connect_RCl, etc.

4.3.34 PID_GROUP_TELEGR_RATE_LIMIT_TIME_BASE (PID = 85)

• Property name:	group telegram rate limitation time base
• Property Datatype:	PDT_UNSIGNED_INT
• Datapoint Type:	None.

The "group telegram rate limitation time base" shall be a 2 octet unsigned value. Its unit shall be 1 ms. So time bases from 1 ms to 65 535 ms can be encoded.

EXAMPLE A time base value of 10 000 shall indicate a time base of 10 s.

4.3.34.1 Usage by the Management Server (device)

Within "group telegram rate limitation time base" the device shall at maximum transmit a number of frames equal to the "group telegram rate limitation number of telegrams". If the "group telegram rate limitation number of telegrams" is set to 1 the time base shall indicate the minimum interframe time between any two group telegrams.

If the time base value is set to zero then the telegram rate limitation shall be disabled independently of the value of "group telegram rate limitation number of telegrams".

Table 20 - Group telegram rate limitation time base values

Value	Meaning
0	Group telegram rate limitation shall be disabled. The system shall not limit the number of telegrams that may be transmitted within any period.
1 to 65 535	Time Base Value multiplied with 1 ms

PID_GROUP_TELEGR_RATE_LIMIT_TIME_BASE shall always be implemented together with PID_GROUP_TELEGR_RATE_LIMIT_NO_OF_TELEGR. For further specifications and requirements, please refer to clause 4.3.35.

4.3.34.2 Usage by the Management Client

Please refer to clause 4.3.35.

4.3.35 PID_GROUP_TELEGR_RATE_LIMIT_NO_OF_TELEGR (PID = 86)

- Property name: group telegram rate limitation number of telegrams
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

4.3.35.1 Usage by the Management Server (device)

The "telegram rate limitation number of telegrams" shall be a 2 octet unsigned value for the number of telegrams that shall be sent at maximum by a device within "telegram rate limitation time base".

Valid data for "telegram rate limitation number of telegrams" are 0 to 65 535.

If the number of telegrams value is set to zero telegram rate limitation shall be disabled independently from the "telegram rate limitation time base" value.

Table 21 - Group telegram rate limitation number of telegrams values

Value	Meaning
0	Telegram rate limitation is disabled. The system shall not limit the number of telegrams that may be transmitted within any period
1 to 65 535	Number of group telegrams to be transmitted at maximum within "telegram rate limitation time base".

Group telegram rate limitation functionality via Properties is an optional feature for devices. But if implemented both Properties "group telegram rate limitation time base" and "group telegram rate limitation number of telegrams" shall be implemented.

Both Properties shall be readable and at least one of them shall be writeable.
If "group telegram rate limitation number of telegrams" is set to 1 the interframe time mechanism shall be active. If "group telegram rate limitation number of telegrams" is greater than 1 the number of group telegrams within the time base shall be counted and limited.

4.3.35.2 Usage by the Management Client

Normal conditions

The Properties “group telegram rate limitation time base” and “group telegram rate limitation number of telegrams” shall be accessed through the Property mechanisms DMP_InterfaceObjectWrite_R, DMP_InterfaceObjectVerify_R and DMP_InterfaceObjectRead_R.

Management telegrams are not limited by the group telegram rate limitation functionality.

Error and exception handling

If the Management Server does not support the group telegram rate limitation Properties, Property description read access will fail.

In case the write access to the Properties “group telegram rate limitation time base” or “group telegram rate limitation number of telegrams” fails, then the Management Client shall skip these accesses and proceed with the rest of the download procedure.

4.3.36 PID_CHANNEL_01_PARAM (PID = 101) to PID_CHANNEL_32_PARAM (PID = 132)

• Property name:	“E-Mode parameters for channel 01” to “E-Mode parameters for channel 32”
• Property Datatype:	PDT_GENERIC_01[]
• Datapoint Type:	None.

These Properties of the Device Object shall provide the parameter blocks for Channel 1 to 32 of the E-Mode Device. The value shall be an array of octets. Its content is defined in the specification of the Channel by the Application Groups. The actual length is static, and depends on the corresponding Channel Code.

Error handling

A negative response (no data) shall be sent each time one of following events appears in the request:

- unknown channel (unknown Property_identifier, channel_number $\geq$ number of channels), or
- unknown parameter (start_index $\geq$ param_block size), or
- incompatible size (start_index+nb_of_elements > param_block size +1).

4.4 Router Object (Object Type 6)

4.4.1 General requirements

For the Coupler, all Properties of this Interface Object shall be used both in Line Coupler mode as well as in Backbone Coupler mode and as well in TP1 Bridge/Repeater Mode.

4.4.2 PID_LOAD_STATE_CONTROL (PID = 5)

The Load State Machine in the Router Object shall control the access to the standard Routing Table.

PID_LOAD_STATE_CONTROL in the Router Object shall support the Load Events as listed in Table 22. For the specification and numerical values of these Load Events, please refer to clause 4.17.2 “Load State Machine – Realisation Type 1 (Property based)”.

Table 22 – Load Events for PID_LOAD_STATE_CONTROL in the Router Object

Load Events
No operation
Start Loading
Load Completed
Unload

The Load States as specified in Table 38 may be returned. For the specification and numerical values of these Load Events, please refer to clause 4.17.2 “Load State Machine – Realisation Type 1 (Property based)”.

Table 23 – Load States returned by PID_LOAD_STATE_CONTROL in the Router Object

Load State
Unloaded
Loaded
Loading
Error

If the Group Address Routing Table is to be checked on received frames and the Load State is not *Loaded* at the same time, no frames shall be routed.

Upon the event *Unload* the State Machine shall transit to the state *Unloaded*. It shall not be relied on the fact that the Routing Table is erased in memory. PID_ROUTETABLE_CONTROL serves to erase the Routing Table in memory.

The default Load State shall be *Loaded* and the Routing Table cleared.

4.4.3 PID_LINE_STATUS (PID = 51)

This Property shall include the status of the Subnetwork connected to the secondary side of the Coupler.

Bit	Name	Description	Coding
0	POWER_DOWN_SUBLINE	Report a power down in the Subnetwork at the secondary side.	0 = power up 1 = power down
1 to 7		reserved	Shall be 0.

The Coupler shall distribute the value of PID_LINE_STATUS to the Subnetwork connected to its primary side with the Management Procedure NM_NetworkParameter_Write_R under the control of the field EN_SUBLINE_STAUS of PID_COUPL_SERV_CONTROL, with Object Type= Router Object and PID = PID_LINE_STATUS in broadcast communication mode; this transmission shall be triggered by a power down or a power restart on the secondary side once only.

```
    /* Announce the status of the Subnetwork on the secondary side */
    /* to the Subnetwork(s) connected to the primary side */
if PID_COUPL_SERV_CONTROL.EN_SUBLINE_STATUS = enable then
    NM_NetworkParameter_Write_R(ASAP = void, comm_mode = point-to-all-points connectionless,
    hop_count_type_req = 6, object_type = Router Object, PID = PID_LINE_STATUS,
    value = PID_LINE_STATUS.Value)
```

4.4.4 PID_MAIN_LCCONFIG (PID = 52) PID_SUB_LCCONFIG (PID = 53)

Two Properties (PID_MAIN_LCCONFIG for the Subnetwork on the primary side and PID_SUB_LCCONFIG for the Subnetwork on the secondary side) shall affect the handling of frames in point-to-point (connectionless and connection oriented) or broadcast communication mode.

Bit	Name	Description	Encoding (d = default value)
0-1	PHYS_FRAME	Specifies the static handling for receiving frames in point-to-point connectionless or connection-oriented communication mode	0 = not used 1 = PHYS_UNLOCK: all frames in point-to-point connectionless and point-to-point connection-oriented communication mode shall be routed, **independent of the Coupler Mode!** 2 = PHYS_LOCK: no frames in point-to-point connectionless and point-to-point connection-oriented communication mode shall be routed, **independent of the Coupler Mode!** 3 = PHYS_ROUT: routing of frames in point-to-point connectionless and point-to-point connection-oriented communication mode shall depend on Destination Address, the Individual Address of the Coupler and the Coupler Mode (normal operation mode) (d)
2	PHYS_REPEAT	Repetition of frames in point-to-point connectionless or connection-oriented communication mode in case of transmission errors.	**Independent of the Coupler Mode!** 0 = no repetitions 1 = frames in point-to-point connectionless or connection-oriented communication mode will be repeated in case of transmission errors (up to 6 times for BUSY acknowledged frames and up to 3 times for other acknowledged or not acknowledged frames) (d)

Bit	Name	Description	Encoding (d = default value)
3	BROADCAST_LOCK	switch ON or OFF the routing of frames in broadcast communication mode	**independent of the Coupler Mode:** 0 = normal: frames in broadcast communication mode will be routed (d) 1 = frames in broadcast communication mode will be blocked (useable during system configuration)
4	BROADCAST_REPEAT	repetition of frames in broadcast communication mode in case of transmission errors	**independent of the Coupler Mode:** 0 = no repetition 1 = frames in broadcast communication mode will be repeated in case of transmission errors (up to 6 times for BUSY acknowledged frames and up to 3 times for other acknowledged or not acknowledged frames) (d)
5	GROUP_IACK_ROUT	specifies the Layer-2 acknowledge of received frames in multicast communication mode. Switch ON or OFF the Layer-2 acknowledge.	**independent of the Coupler Mode:** 0 = all multicast frames will be acknowledged, independently of the routing! (useful only to avoid the repetitions of misrouted frames) 1 = normal mode (all multicast frames that will be routed will also be acknowledged) (d)
6-7	PHYS_IACK	specifies the Layer-2 acknowledge of received frames in point-to-point connectionless or – connection-oriented communication mode. Switch ON or OFF the Layer-2 acknowledge.	**independent of the Coupler Mode:** 0 = not used 1 = normal mode (all frames that will be routed or that are addressed to the Coupler itself will be acknowledged) (d) 2 = all frames will be acknowledged (useful only to avoid the repetitions of misrouted frames) 3 = all frames on point-to-point connectionless – or connection-oriented communication mode shall be negatively acknowledge (NACK). This shall serve for protection purposes. (It is useful to prevent all parameterisation in one Subnetwork; the Coupler shall be protected too. A typical use case is the protection of a Subnetwork that is located outside a building)

4.4.5 PID_MAIN_LCGRPCONFIG (PID = 54) PID_SUB_LCGRPCONFIG (PID = 55)

Two Properties (PID_MAIN_LCGRPCONFIG for the Subnetwork on the primary side and PID_SUB_LCGRPCONFIG for the Subnetwork on the secondary side) shall affect the handling of frames in multicast communication mode.

These values shall be interpreted independently of the Coupler Mode.

Bit	Name	Description	Encoding (d = default value)
0-1	GROUP_6FFF	specify the handling of standard Group Addressed frames with Group Address ≤ 6FFFh	0 = not used 1 = GROUP_UNLOCK6FFF All standard Group Addresses frames shall be routed. **(Typical for TP1 Bridge/Repeater Mode.)** 2 = GROUP_LOCK6FFF No standard Group Addressed frames shall be routed. 3 = GROUP_ROUT6FFF The Routing of the standard Group Addressed frames shall depend on the Routing Table (d).
2-3	GROUP_7000	specify the handling of group addressed frames ≥ 7000h	0 = not used 1 = GROUP_UNLOCK7000 All frames shall be routed. **(Typical for TP1 Bridge/Repeater Mode.) (d)** 2 = GROUP_LOCK7000 No frames shall be routed. 3 = GROUP_ROUT7000 Routing shall depend on the Routing Table.

Bit	Name	Description	Encoding (d = default value)
4	GROUP_REPEAT	repetition of group addressed frames in case of transmission errors	0 = No repetition. 1 = Frames shall be repeated in case of transmission errors (up to 6 times for BUSY acknowledged frames and up to 3 times for other acknowledged or not acknowledged frames). **(d)**
5-7		not used	Shall be 0.

4.4.6 PID_ROUTETABLE_CONTROL (PID = 56)

4.4.6.1 Structure

This Property shall be used for the Management of the Routing Table for standard KNX Group Addresses. Due to the Property Datatype PDT_Function the configuration is much faster and more flexible than the standard access to a fixed routing address field. The structure of the Property is defined in Figure 5.

octet 10	octet 11	octet 12	octet 13 to octet 21
00h or return_code	ServiceID	ServiceInfo1	ServiceInfo2 ... ServiceInfo11

Figure 5 - Structure of the Function Property for reading or writing (octet numbering for TP1)

The ServiceID shall specify the selected method. For some methods additional ServiceInfos shall be necessary.

4.4.6.2 ServiceID: 1 (SRVID_CLEAR_ROUTINGTABLE)

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11
00h	01h

For SRVID_CLEAR_ROUTINGTABLE no additional ServiceInfo fields shall be used.

The Management Client shall apply SRVID_CLEAR_ROUTINGTABLE with A_FunctionPropertyCommand-PDU to clear all standard Group Addresses in the Routing Table.

The Coupler shall on reception of SRVID_CLEAR_ROUTINGTABLE clear all standard Group Addresses in the Routing Table and shall respond with an A_FunctionPropertyState_Response-PDU as follows.

Response (A_PropertyPropertyState_Response-PDU)

octet 10	octet 11
return_code	01h

The Coupler shall use the following values as return_code.

return_code:	00h:	clearing of the Routing Table has been successful.
	FFh:	an error has occurred during the clearing of the Routing Table (verify error)

b) Read (A_FunctionPropertyState_Read-PDU)

The Management Client shall apply the SRVID_CLEAR_ROUTINGTABLE with A_FunctionProperty-State_Read-PDU to check if all standard Group Addresses are cleared in the Routing Table.

The Coupler shall use the following values as return_code.

return_code:	00h:	all standard Group Addresses in the Routing Table are cleared.
	FFh:	not all standard Group Addresses in the Routing Table are cleared.

4.4.6.3 ServiceID: 2 (SRVID_SET_ROUTINGTABLE)

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11
00h	02h

For SRVID_SET_ROUTINGTABLE no additional ServiceInfo fields shall be used.

The Management Client shall apply SRVID_SET_ROUTINGTABLE with A_FunctionPropertyCommand-PDU to set all standard Group Addresses in the Routing Table.

The Coupler shall on reception of SRVID_SET_ROUTINGTABLE set all standard Group Addresses in the Routing Table and shall respond with an A_FunctionPropertyState_Response_PDU as follows.

Response (A_PropertyPropertyState_Response-PDU)

octet 10	octet 11
return_code	02h

The Coupler shall use the following values as return_code.

return_code:	00h:	setting of the Routing Table has been successful.
	FFh:	an error has occurred during the setting of the Routing Table (verify error)

b) Read (A_FunctionPropertyState_Read-PDU)

The Management Client shall apply the SRVID_SET_ROUTINGTABLE with A_FunctionProperty-State_Read-PDU to check if all standard Group Addresses are set in the Routing Table.

The Coupler shall use the following values as return_code.

return_code:	00h:	all standard Group Addresses in the Routing Table set.
	FFh:	not all standard Group Addresses in the Routing Table are set.

4.4.6.4 ServiceID: 3 (SRVID_CLEAR_GROUPADDRESS)

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
00h	03h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Management Client shall apply SRVID_CLEAR_GROUPADDRESS to clear one or more standard Group Addresses in the Routing Table of the Coupler.

START ADDRESS	This shall be the first standard Group Address that shall be cleared.
END ADDRESS	This shall be the last standard Group Address that shall be cleared.

If only one standard Group Address is to be modified then START ADDRESS and ENDADDRESS shall be equal!

If START ADDRESS is greater than END ADDRESS no address will be modified (error response!)

The Coupler shall on reception of SRVID_CLEAR_GROUPADDRESS clear all Standard Group Addresses in the Routing Table starting from START ADDRESS up to and including END ADDRESS. It shall respond with an A_FunctionPropertyState_Response-PDU as follows.

Response (A_PropertyPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
return_code	03h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Coupler shall use the following values as return_code.

return_code:	00h:	The standard Group Addresses are cleared successfully.
	FFh:	One or more errors have occurred during the clearing of the standard Group Addresses (verify error).

b) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
00h	03h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Management Client shall apply the SRVID_CLEAR_GROUPADDRESS with A_FunctionProperty-State_Read-PDU to check if all standard Group Addresses in the range from START ADDRESS to END ADDRESS are cleared.

START ADDRESS	This shall be the first standard Group Address that shall be checked.
END ADDRESS	This shall be the last standard Group Address that shall be checked.

If only one standard Group Address shall be checked then START ADDRESS and END ADDRESS shall be equal!

If START ADDRESS is greater than END ADDRESS then no standard Group Address shall be checked.

Response (A_PropertyPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
return_code	03h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Coupler shall use the following values as return_code.

return_code:	00h:	all checked standard Group Addresses are cleared.
	FFh:	not all checked standard Group Addresses are cleared.

4.4.6.5 ServiceID: 4 (SRVID_SET_GROUPADDRESS)

a) Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
00h	04h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Management Client shall apply SRVID_SET_GROUPADDRESS to set one or more standard Group Addresses in the Routing Table of the Coupler.

START ADDRESS	This shall be the first standard Group Address that shall be set.
END ADDRESS	This shall be the last standard Group Address that shall be set.

If only one standard Group Address is to be modified then START ADDRESS and END ADDRESS shall be equal!

If START ADDRESS is greater than END ADDRESS no standard Group Address shall be modified (error response!).

The Coupler shall on reception of SRVID_SET_GROUPADDRESS set all Standard Group Addresses in the Routing Table starting from START ADDRESS up to and including END ADDRESS. It shall respond with an A_FunctionPropertyState_Response-PDU as follows.

Response (A_PropertyPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
return_code	04h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Coupler shall use the following values as return_code.

return_code:	00h:	The standard Group Addresses are set successfully.
	FFh:	One or more errors have occurred during the setting of the standard Group Addresses (verify error).

b) Read (A_FunctionPropertyState_Read-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
00h	04h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Management Client shall apply the SRVID_SET_GROUPADDRESS with A_FunctionProperty-State_Read-PDU to check if all standard Group Addresses in the range from START ADDRESS to END ADDRESS are set.

START ADDRESS	This shall be the first standard Group Address that shall be checked.
END ADDRESS	This shall be the last standard Group Address that shall be checked.

If only one standard Group Address shall be checked then START ADDRESS and END ADDRESS shall be equal!

If START ADDRESS is greater than END ADDRESS then no standard Group Address shall be checked.

Response (A_PropertyPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15
return_code	04h	START ADDRESS		END ADDRESS	
		high octet	low octet	high octet	low octet

The Coupler shall use the following values as return_code.

return_code:	00h:	all checked standard Group Addresses are set.
	FFh:	not all checked standard Group Addresses are set.

4.4.6.6 Error handling

The reaction to an unknown ServiceID shall be as follows

a) Write (A_FunctionPropertyCommand-PDU), A_PropertyValue_Read-PDU)

octet 10	octet 11	…
00h	undefined ServiceID	…

b) response A_PropertyValue_Response-PDU

octet 10	octet 11
return_code	undefined ServiceID

The Coupler shall use the following value as return_code.

return_code: FFh: error.

4.4.7 PID_COUPL_SERV_CONTROL (PID = 57)

The Property PID_COUPL_SERV_CONTROL in the Router Object (object_type = 6) shall be used to control SNA management in a Coupler.

Bit	Name	Description	Encoding (d = default value)
0	EN_SNA_INCONSISTENCY_CHECK	This bit shall enable and disable the inconsistency-check of frames in point-to-point connectionless and connection-oriented communication mode. This feature shall only be executed if the Coupler is not in TP1 Bridge/Repeater Mode! See further details below.	0 = disable (d) 1 = enable
1	EN_SNA_HEARTBEAT	This bit shall enable and disable “SNA heartbeat” (see [09] clause 1.3.6 “SNA heartbeat”) in the Subnetwork connected to the Coupler’s secondary side. This feature shall only be executed if the Coupler is not in TP1 Bridge/Repeater Mode!	0 = disable (d) 1 = enable

Bit	Name	Description	Encoding (d = default value)
2	EN_SNA_UPDATE_WRITE	This bit shall enable and disable the update of the Subnetwork Address by the Coupler in the Subnetwork Address connected to its secondary side by an NM_SubnetworkAddress_Write_Local-Subnetwork (see [08]) after its own Individual Address has changed via programming mode or KNX Serial Number. This feature shall only be executed if the Coupler is not in TP1 Bridge/Repeater Mode!	1 = enable 0 = disable (d)
3	EN_SNA_READ	This bit shall enable and disable the server side support of the SNA read procedure in the Coupler (see [09]). If this feature is enabled, the Coupler shall accept the A_NetworkParameter_Read-service in that Management Procedure and respond as specified in [09]. This feature shall only be executed if the Coupler is not in TP1 Bridge/Repeater Mode! If the Coupler is in TP1 Bridge/Repeater Mode then this procedure shall always be ignored, independently of the Property setting. See further details below.	0 = disable 1 = enable (d)
4	EN_SUBLINE_STATUS	This bit shall enable or disable the distribution of PID_LINE_STATUS in the Subnetwork connected to the primary side on power failure and power return in the Subnetwork Connected to the secondary side. This feature shall be executed independent of TP1 Bridge/Repeater Mode!	0 = disable (d) 1 = enable
5 to 7	reserved		Shall be 0.

EN_SNA_INCONSISTENCY_CHECK

A Router may detect topologically incorrect Individual Addresses on its primary side and on its secondary side and perform the procedure “SNA update on SNA inconsistency”. The field EN_SNA_INCONSISTENCY_CHECK shall control whether (enable) or not (disable) the Router shall detect and correct incorrect Subnetwork Addresses. Please refer to [09] for the specifications.

EN_SNA_READ (indication)

Newly installed devices may perform an “SNA Read” (see [09]) to get the current SNA. The field EN_SNA_READ shall control whether (enable) or not (disable) the Router shall react to this procedure or not. Please refer to [09] for the specifications.

4.4.8 PID_MAX_APDU_LENGTH (PID = 58)

- Property name: MAX. APDU-Length
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: None.

4.4.8.1 Usage by the Management Client

The Management Client shall interpret PID_MAX_APDU_LENGTH in the Router Object as the APDU-length of the frames that the Router or Coupler can route.

If this Property is not present in the Router Object, the Management Client shall evaluate the same property PID_MAX_APDU_LENGTH (PID = 56) in the Device Object (see clause 4.3.7).

EXAMPLE If a Management Client needs to find out the maximal APDU-length that it can use to manage (download) a target device (Management Server) then it also needs to know the maximal APDU-length that is supported by any possible in-between Router. To this, is shall read PID_MAX_APDU_LENGTH in the Router Object of these Routers. This is specified in [09] under “Discovery of maximal frame length” Step 3.

4.4.9 PID_L2_COUPLER_TYPE (PID = 59)

- Property name: L2 Coupler Type
- Property Datatype: PDT_BITSET8
- Datapoint Type: None.

4.4.9.1 Format

The Property *L2 Coupler Type* shall be an 8 bit bitset.

Bit	Name	Value	Meaning
0	IS_REPEATER	0	Device is a TP1 Bridge
		1	Device is a TP1 Repeater
1 to 7	Reserved	0	These bits are reserved for future extensions.

The default value shall be “TP1 Repeater” (01h).

4.4.9.2 Usage by the Management Server (device)

Bit 0

The Property *L2 Coupler Type* shall not be evaluated if the device is a Router, this is, if the Device Address equals 0.

If the Device Address differs from 0 and Property *L2 Coupler Type* is not implemented, then the Coupler shall be a TP1 Repeater.

If the Device Address differs from 0 and Property *L2 Coupler Type* is implemented then the Coupler shall be a TP1 Bridge if bit 0 equals 0 and a TP1 Repeater is bit 0 equals 1.

This is summarized in the following pseudo-code.

```
if Device Address = 0
    then Device is Router
else
    if PID_L2_COUPLER_TYPE.BIT0 = 0
        then Device is TP1 Bridge
        else Device is TP1 Repeater
    endif
endif
```

Bit 1 to bit 7

The value of the reserved bits (bit 1 to bit 7) shall be ignored by the Management Server, even if they are set to 1. In the A_PropertyValue_Response-PDU, confirming the A_PropertyValue_Write-PDU back to the Management Client, the Management Server (device) shall clear these bits.

4.4.9.3 Usage by the Management Client

4.4.9.3.1 Read access

Bit 0

The Management Client shall interpret this Property Value, in combination with the Device Address of the Management Server, identically as the Management Server, as specified in 4.4.9.2 above. This shall allow the Management Client to determine whether the device is a TP1 Repeater or a TP1 Bridge. If this Property is not present, then the device shall be considered as a TP1 Repeater.

Bit 1 to bit 7

The Management Client shall ignore the value of these bits.

4.4.9.3.2 Write access

Inputs

- Constants

Bit 1 to bit 7:	The Management Client shall clear these bits.

- Application specific

  None

- From user

Bit 0	The value of PID_L2_COUPLER_TYPE.BIT0 shall be provided by the user of the Management Client.

- From the Management Server (device)

  None.

Access

The Management Client shall set the value of Property PID_L2_COUPLER_TYPE using the Procedure DMP_InterfaceObject_Write_R.

4.5 LTE Address Routing Table Object (Object Type 7)

4.5.1 General requirements

This Interface Object shall exclusively control the handling of LTE-HEE extended address type frames.

Please refer to [15] for the specification of which Properties are mandatory or optional in the LTE Address Routing Table Object.

4.5.2 PID_LOAD_STATE_CONTROL (PID = 5)

The Load State Machine in the LTE Address Routing Table Object shall control the access to the LTE Address Routing Table.

PID_LOAD_STATE_CONTROL in the LTE Address Routing Table Object shall support the Load Events as listed in Table 24. For the specification and numerical values of these Load Events, please refer to clause 4.17.2 “Load State Machine – Realisation Type 1 (Property based)”.

Table 24 – Load Events for PID_LOAD_STATE_CONTROl in the LTE Address Routing Table Object

Load Events
No operation
Start Loading
Load Completed
Unload

The Load States as specified in Table 25 may be returned. For the specification and numerical values of these Load Events, please refer to clause 4.17.2 “Load State Machine – Realisation Type 1 (Property based)”.

Table 25 – Load States returned by PID_LOAD_STATE_CONTROL in the LTE Address Routing Table Object

Load State
Unloaded
Loaded
Loading
Error

If the LTE Address Routing Table is to be checked on received frames and the Load State of the LTE Address Routing Table Object is not *Loaded* at the same time, no LTE-HEE extended address type frames shall be routed.

The default Load State shall be *Loaded*. The LTE Address Routing Table shall be cleared. That means the current length shall be zero and all LTE-HEE extended address type frames shall be routed.

4.5.3 PID_LTE_ROUTESELECT (PID = 51)

Bit	Name	Description	Encoding (d = default value)
0-1	LTESUB	Shall specify the general handling for LTE-HEE extended address type frames from the Subnetwork on the secondary side	0 = not used 1 = LTESUB_PASS All frames shall be routed. 2 = LTESUB_BLOCK No frames shall be routed. 3 = LTESUB_NORMAL Routing shall depend on the LTE Address Routing Table (d).
2-3		reserved	Shall be 0.
4-5	LTEMAIN	Shall specify the general handling for LTE-HEE extended address type frames from the Subnetwork on the primary side.	0 = not used 1 = LTEMAIN_PASS All frames shall be routed. 2 = LTEMAIN_BLOCK No frames shall be routed. 3 = LTEMAIN_NORMAL Routing shall depend on the LTE Address Routing Table (d).
6-7		reserved	Shall be 0.

4.5.4 PID_LTE_ROUTETABLE (PID =52)

- Property name: LTE Address Routing Table Object
- Property Datatype: PDT_GENERIC_05
- Datapoint Type: None

The LTE Address Routing Table shall only be checked if the Property PID_LTE_ROUTESELECT is set to LTESUB_NORMAL and LTEMAIN_NORMAL. The table shall be a PDT_GENERIC_05 array with a maximum of 32 entries. If the current length is zero all for LTE-HEE extended address type frames shall be routed.

Attention

It shall not be forgotten to handle the LTE state machine during the modification of the Routing Table!

octet 1								octet 2	octet 3	octet 4	octet 5
Direction				EFF				Base		Mask	
b7	b6	b5	b4	b3	b2	b1	b0	high octet	low octet	high octet	low octet
0	0	0	0	0	1	x	x				
0	0	0	0	0	1	x	x				

- Octet 1:
  - b0 – b3: EFF (Extended Frame Format)
  - b4 and b5: reserved (fixed to 0)
  - b6: data direction: secondary side to primary side free (0) / blocked (1)
  - b7: data direction: primary side to secondary side free (0) / blocked (1)

- Octet 2 und 3: „Base address“
- Octet 4 und 5: „Mask“

4.6 cEMI Server Object (Object Type 8)

4.6.1 PID_MEDIUM_TYPE (PID = 51)

• Property name:	Medium Type
• Property Datatype:	PDT_BITSET16
• Datapoint Type:	DPT_Media

This Property shall contain the Media Type(s) supported by cEMI server.

The format complies with the encoding of DPT_Media as specified in [12].

NOTE 12 A cEMI server could be connected to more than one medium 8); e.g. when the cEMI server device is also a Media Coupler.

4.6.2 PID_COMM_MODE (PID = 52)

• Property name:	Communication Mode
• Property Datatype:	PDT_ENUM8
• Datapoint Type:	DPT_CommMode (DPT_ID = 20.1000)

Please refer to [11] for the handling and interpretation of PID_COMM_MODE by the cEMI Server.

4.6.3 PID_MEDIUM_AVAILABILITY (PID = 53)

• Property name:	Medium Availability
• Property Datatype:	PDT_BITSET16
• Datapoint Type:	DPT_Media (DPT_ID = 22.1000)

The Property PID_MEDIUM_AVAILABILITY shall indicate whether the connected medium is active (availability is true = 1) or not (availability is false = 0).

The same Datapoint Type shall be used as for the Property PID_MEDIUM_TYPE. It shall be a 2 octet long bit set B_{16}, in which each bit shall indicate if the corresponding medium is active or not. Please also refer also to 3.2.3.2 “Supported Media Type”.

4.6.4 PID_ADD_INFO_TYPES (PID = 54)

• Property name:	Additional Information Types
• Property Datatype:	PDT_ENUM8[]
• Datapoint Type:	DPT_AddInfoTypes (DPT_ID = 20.1001)

The Property PID_ADD_INFO_TYPES shall indicate which Additional Information Types the cEMI server supports. If the Property is not present, the cEMI server shall not support any Additional Information Types.

An array-structured Property shall be used to represent the supported Additional Information Types. Each (used) array field shall represent one supported Additional Information Type. The Property shall contain as much array fields as different Additional Information Types are supported. The number of used array fields (= number of supported Additional Information Types) can be read (read only) as array-element with start index 0. Within the array, the type IDs shall be sorted in ascending order.

8) Such a device does not exist yet (May-2002). For separation to which medium a *L_Data.req* message shall be sent, respectively from which medium a *L_Data.ind* message was received, a ‘channel number’ could be defined as a further ‘additional information’.

The format of PID_ADD_INFO_TYPES (N_8) shall comply with the encoding of DPT_AddInfoTypes as specified in [12].

The codes *DPT_AddInfoType* shall be the same as specified in clause “Additional information” in [11].

4.6.5 PID_TIME_BASE (PID = 55)

- Property name: Time Base
- Property Datatype: PDT_UNSIGNED_INT
- Datapoint Type: DPT_Value_2_Ucount (DPT-ID = 7.001)

The Property PID_TIME_BASE shall contain the time base used by the cEMI Server for the extended relative timestamp. The value shall be the length of one tick of the free running counter used in the extended relative timestamp in nanoseconds.

Please refer to [11] for the requirements concerning the use of PID_TIME_BASE in the cEMI Server Object.

4.6.6 PID_TRANSP_ENABLE (PID = 56)

- Property name: cEMI Server Link Layer Transparency Control
- Property Datatype: PDT_BINARY_INFORMATION
- Datapoint Type: DPT_Enable (DPT_ID = 1.003)

The Property PID_TRANSP_ENABLE shall be used to select the Data Link Layer Transparency Mode of the cEMI server between “Normal Transparency” and “Full Transparency”.

Mode	Value	Specification
“Normal Transparency”	0	In this mode, the cEMI frames shall all be sent onto the bus medium with the cEMI server device’s own Individual Address as Source Address.
“Full Transparency”	1	In this mode, the cEMI client shall have an own Individual Address, that can be different from the cEMI server device’s Individual Address. In Full Transparency mode, the cEMI frames shall all be sent onto the bus medium with the cEMI client’s Individual Address as Source Address.

The default value shall be “Normal Transparency”. If the Property PID_TRANSP_ENABLE is not present, the Transparency Mode shall be “Normal Transparency” as well.

NOTE 13 On TP media (and other media with time critical Data Link Layer acknowledge mechanisms), ‘full transparency’ mode is not applicable.

4.6.7 PID_BIBAT_NEXTBLOCK (PID = 59)

- Property name: BiBat Next Block
- Property Datatype: PDT_UNSIGNED_CHAR
- Datapoint Type: DPT_Value_1_Ucount (5.010)

This Property is mandatory if the cEMI Server is a BiBat Master.

The cEMI Server that has BiBat Master functionality shall spontaneously send an M_PropInfo.ind message to the cEMI client with the value of PID_BIBAT_NEXT_BLOCK. It shall contain the number of the next BiBat block in the Master timeslot scheme. The allowed range is from 0 to 127.

Refer to [11] clause 4.1.5.5.2 “L_Data.req for BiBat synchronous data frames” as well.

4.6.8 PID_RF_MODE_SELECT (PID = 60)

- Property name: RF Communication Mode
- Property Datatype: PDT_ENUM8 (alt: PDT_UNSIGNED_CHAR)
- Datapoint Type: None.

With this Property the RF communication mode of the cEMI Server can be selected.

0: asynchronous (mandatory)

1: asynchronous + BiBat Master (optional)

2: asynchronous + BiBat Slave (optional)

Write access to this Property is optional if only asynchronous RF communication mode is supported.

4.6.9 PID_RF_MODE_SUPPORT (PID = 61)

- Property name: Supported RF Communication Modes
- Property Datatype: PDT_BITSET8 (alt.: PDT_GENERIC_01)
- Datapoint Type: None.

With this Property the supported RF communication mode(s) of the cEMI Server can be identified by the cEMI Client and accordingly the mode to be used can be selected.

Field	**b_7**	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
Name	reserved	reserved	reserved	reserved	reserved	BiBat Slave	BiBat Master	Asynch
Values	0	0	0	0	0	0, 1	0, 1	1

Bit	**Data fields**	**Description**	**Encoding**	**Unit**	**Range**
b_0	Asynchronous	asynchronous mode support	(0 = value not allowed) 1 = true	none	{0,1}
b_1	BiBat Master	BiBat Master mode supported	0 = false 1 = true	none	{0,1}
b_2	BiBat Slave	BiBat Slave mode supported	0 = false 1 = true	none	{0,1}
$b_3 \ldots b_7$	reserved	reserved, shall be 0	not applicable	n.a.	n.a.

This Property shall be read only.

4.6.10 PID_RF_FILTERING_MODE_SELECT (PID = 62)

- Property name: RF Filtering Mode Selection
- Property Datatype: PDT_ENUM8 (alt: PDT_UNSIGNED_CHAR)
- Datapoint Type: None.

With this Property the RF filtering mode for received RF frames in the cEMI Server shall be selected.

0: no filtering, all supported received frames shall be passed to the cEMI Client using L_Data.ind
1: filtering by Domain Address (optional)
2: filtering by KNX Serial Number table (optional)
3: filtering by Domain Address and by KNX Serial number table (optional)

4.6.11 PID_RF_FILTERING_MODE_SUPPORT (PID = 63)

- Property name: Supported RF Filtering Modes
- Property Datatype: PDT_BITSET8 (alt.: PDT_GENERIC_01)
- Datapoint Type: None.

With this Property the supported RF filtering mode(s) of the cEMI Server can be identified by the cEMI Client and accordingly the filtering mode to be used can be selected.

Field	**b_7**	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
Name	reserved	reserved	reserved	reserved	reserved	Filter by DoA & SerialNr	Filter by SerialNr	Filter by DoA
Value	0	0	0	0	0	0/1	0/1	0/1

This Property shall be read only.

4.7 KNXnet/IP Parameter Object (Object Type 11)

For the functionality of the KNXnet/IP Parameter Object and the specification of its Properties, please refer to [14].

4.8 File Server Object (Object Type 13)

4.8.1 Overview

Table 26 – File Server Object

Property Name	Property Identifier	PDT/DPT	Description
Interface Object type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	13 = File Server Object
MAX. APDU-Length	51 = PID_MAX_APDU_LENGTH_OUT	PDT_UNSIGNED_INT	Maximal APDU-length that shall be used during a file transfer.
File Command	52 = PID_FILE_COMMAND	PDT_FUNCTION	Command
File Path	53 = PID_FILE_PATH	PDT_UTF-8[] (alt. PDT_UNSIGNED_CHAR[])	Path to the directory or file
MAX. APDU-Length	56 = PID_MAX_APDU_LENGTH	PDT_UNSIGNED_INT	Maximal APDU-length supported by the File Server.

[a] This Property is mandatory if long frames are supported.

4.8.2 Property MAX. APDU-Length Out (PID = 51)

This Property shall be an optional writable Property and shall provide the maximum APDU-Length that shall be used by this File Server Object for the A_FileStream_InfoReport-service. This Property is mandatory if Long Frames are supported. With this Property a client can adjust the maximum used APDU length for an FTP procedure.

If this Property is not implemented then the used maximal APDU-length shall be 14.

4.8.3 Property File Command (PID = 52)

4.8.3.1 Definition

This Property shall be mandatory and shall be formatted as a PDT_FUNCTION i.e. the number of octets written shall be variable and the number of octets responded shall also be variable. Each command is described in detail below. The interpretation of the Return Code (Error Code) depends on the given Command. A read request to this Property shall always return the command result of the preceding command or “Command successful” if there has been no preceding command.

Prior to any File Command or access to any other Property of the File Server Object, a File Client shall at first request a File Handle from the File Server through the command “Get File Handle”. Only if a valid File Handle is returned, the File Client may use the current File Server Object.

The responded Return Code (Error Code) depends on the given command; these are specified for the different commands below.

4.8.3.2 Overview Return Codes

Table 27 – Summary of command independent Return Codes

Return Code	Code Name	Description
00h	Command successful	This is the general return code to indicate a successful command execution.
01h to 0Fh	Reserved	These Return Codes are reserved for future use and shall not be used.
10h	Invalid command	The preceding command is reserved or not supported by this File Server.
11h to 1Fh	Reserved	These Return Codes are reserved for future use and shall not be used.

Table 28 – Summary of command dependent Return Codes

Return Code	Code Name	Description
20h	Object busy	The File Server indicates that it is busy and cannot accept any file command in this File Server Object.
21h	Command not possible	The given command cannot be executed in the requested way.
22h	Not found	The current path name does not lead to any known file or directory.
23h	Write protected	The specified file or directory is write protected.
24h	Access denied	The specified file or directory is not accessible in the requested way.
25h	No more space available	The File Servers disk or storage memory is full. Incoming data cannot be stored.
26h	Directory not empty	The specified directory is not empty and can therefore not be removed.
27h to FFh	Reserved	These Return Codes are reserved for future use and shall not be used.

4.8.3.3 Command

4.8.3.3.1 Overview of the commands and general requirements

Most commands affect the treatment of a file and shall therefore always be executed with a file path or directory path as a parameter. This parameter shall either be written to the Property PID_FILE_PATH or shall be part of the command. This is specified for every individual command below.

If a file path or directory path is part of the command, the same coding rules shall be applied as defined for the Property File Path. See 4.8.4.

There are commands for an FTP based client, for an HTTP based clients and for both. This has the advantage that the same file can be read in different modes depending on its usage in the client.

EXAMPLE A heading part and a footing part can automatically be added to an HTML file, when it is read with an HTTP based client. This will not be done with an FTP based client.

Every command shall be a single, isolated event, this is, the File Server shall interpret any command without knowledge of any preceding commands. The exception to this rule shall be the command "Rename To"; this command shall only be executed if the preceding commands is a command "Rename From".

Table 29 – File Commands

Command value	Command name	Parameter in other Properties	Mandatory / optional
00h	Reserved		O
01h	Get File Handle		M
02h to 10h	Reserved		O
11h	Retrieve File (ftp)	file path	O
12h	Store File (ftp)	file path	O
13h	List Directory (ftp)	dir path	O
14h	Rename From (ftp)	file path / dir path	O
15h	Rename To (ftp)	file path / dir path	O
16h	Delete (ftp)	file path	O
17h	Remove Directory (ftp)	dir path	O
18h	Make Directory (ftp)	dir path	O
19h to 1Ch	Reserved		O
1Dh	Get File Size (ftp)	file path	O
1Eh	Get Empty Disk Space (ftp)	n/a	O
1Fh	Abort (http, ftp)	n/a	M [a]
20h	Reserved		O
21h	Get File (http)	file path	O
22h	Post File (http)	file path	O
23h to FFh	Reserved		O

[a] This command is mandatory only if one of the following commands are implemented: Retrieve File, Get File, List Directory.

4.8.3.3.2 Command “Get File Handle” (command value 01h)

This command shall cause the File Server to return a new valid File Handle.

Property File Command - Write (A_FunctionPropertyCommand-PDU)

octet 10
Command Value

Property File Command - Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11
Return Code	Return Value

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate a successful request of a File Handle. The **Return Value** shall denote the **File Handle** that shall be used by the File Server to identify the file in the A_FileStream_InfoReport service. File Handles shall be in the range of 0 to 15. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”. NOTE: All values 0 to 15 denote valid File Handles!
20h	Object busy	This Return Code shall be used by the File Server to indicate that it is busy and cannot accept any file command in this File Server Object. The **Return Value** shall denote the **Object Index** of the next free File Server Object. The decision by the File Client whether to try a second File Server Object can be based on this information. An Object Index value of 0 denotes that no other free File Server Object is available. An Object Index value of 1 to 255 is a valid index to a free File Server Object.

NOTE 14 If only one File Server Object is implemented, the Return Value may always be 0 (for the File Handle and for the next free File Server Object Index) and therefore may be a constant value.

4.8.3.3.2.1 Command “Retrieve File” (command value 11h)

This command shall cause the File Server to transfer a copy of the file of which the path is specified to the File Client. The status and contents of the file in the File Server shall be unaffected. For the transmission of the file the Management Procedure FTP_RetrieveFile shall be applied as specified in [08] clause “File Transfer Procedures”., this is, the service A_FileStream_InfoReport shall be called consecutively by the File Server.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command "Retrieve File". The File Server shall handle this request as specified in [08] clause "File Transfer Procedures".
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known file.
24h	Access denied	The File Server shall respond with this Return Code if the file cannot be read. NOTE This may be because the file is currently internally written, because another File Client is currently writing into this file, etc. This is not differentiated any further.

4.8.3.3.2.2 Command "Store File" (command value 12h)

This command shall cause the File Server to accept data from the File Client and store them as a file in its file system. If the file exists, then its contents shall be replaced. A new file shall be created if the file specified in the path name does not already exist. If the new file name is longer than the acceptable file name length, the file system may adjust the new file name automatically.

NOTE 15 This is equivalent to the store unique command in FTP.

For the transmission of the file the Management Procedure FTP_StoreFile shall be applied as specified in in [08] clause "File Transfer Procedures", this is, the service A_FileStream_InfoReport shall be called consecutively by the File Client.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command "Store File". The File Server shall handle this request as specified in in [08] clause "File Transfer Procedures".
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known directory.
23h	Write protected	The specified file is write protected.
24h	Access denied	The File Server shall respond with this Return Code if the file cannot be stored. NOTE 16 This may be because the file is currently internally written, because another File Client is currently writing to the file, etc. This is not differentiated any further.
25h	No more space available	The File Servers disk or storage memory is full. Incoming data cannot be stored.

4.8.3.3.3 Command "List Directory" (command value 13h)

This command shall cause the File Server to send a list of all entries of a directory of which the path is specified, to the client. A definition of the transfer format is given in [07] clause "Directory Listing Format". For the transmission of the directory listing the Management Procedure FTP_ListDirectory shall be applied as specified in [08] clause "File Transfer Procedures", this is, the service A_FileStream_InfoReport shall be called consecutively by the File Server.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	Directory Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “List Directory”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known directory.

4.8.3.3.4 Command “Rename From” (command value 14h)

This command shall cause the File Server to accept the path name as the name of the file to be renamed. It shall be followed by a command “Rename To”. To rename a file the Management Procedure FTP_Rename as specified in [08] clause “File Transfer Procedures” shall be used.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Rename From”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known file or directory.

Return Code	Code Name	Description
23h	Write protected	The specified file is write protected.
24h	Access denied	The File Server shall respond with this Return Code if the file cannot be renamed. NOTE 17 This may be because the file is currently internally accessed in the device, because another File Client is currently accessing the file, etc. This is not differentiated any further.

4.8.3.3.4.1 Command "Rename To" (command value 15h)

This command shall specify the new path name of the file specified in the immediately preceding command "Rename From". Together the two commands shall cause a file to be renamed. To rename a file the Management Procedure FTP_Rename as specified in [08] clause "File Transfer Procedures" shall be used.

If the new file name is longer than the acceptable file name length, the file system may adjust the new file name automatically.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command "Rename To". The File Server shall handle this request as specified in [08] clause "File Transfer Procedures".
10h	Invalid command	The given command is reserved or not supported by this File Server.
21h	Command not possible	The File Server shall respond with this Return Code if no command "Rename From" has preceded.
22h	Not found	The current path name does not lead to any known directory.

4.8.3.3.4.2 Command “Delete” (command value 16h)

This command shall cause the file specified in the path name to be deleted in the file system of the File Server.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Delete”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known file.
23h	Write protected	The specified file is write protected.
24h	Access denied	The File Server shall respond with this Return Code if the file cannot be deleted. NOTE 18 This may be because the file is currently internally accessed in the device, because another File Client is currently accessing the file, etc. This is not differentiated any further.

4.8.3.3.4.3 Command “Remove Directory” (command value 17h)

This command shall cause the directory specified in the path name to be deleted in the file system of the File Server. This command shall by the File Server only be executed if the directory is empty. If the directory contains files or other directories, then the directory shall not be deleted.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	Directory Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Remove Directory”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known directory.
23h	Write protected	The specified directory is write protected.
24h	Access denied	The File Server shall respond with this Return Code if the directory cannot be deleted. NOTE 19 This may be because the directory is currently internally accessed in the device, because another File Client is currently accessing the directory, etc. This is not differentiated any further.
26h	Directory not empty	The specified directory is not empty and can therefore not be removed.

4.8.3.3.4.4 Command “Make Directory” (command value 18h)

This command shall cause the directory specified in the path name to be created in the file system of the File Server. If the new directory name is longer than the acceptable directory name length, the File Server may adjust the new directory name automatically.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	Directory Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Remove Directory”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known parent directory.
23h	Write protected	The specified parent directory is write protected. A new directory cannot be created here.
24h	Access denied	The File Server shall respond with this Return Code if the directory cannot be created due to other reasons.
25h	No more space available	The File Servers disk or storage memory is full. A new directory cannot be created.

4.8.3.3.4.5 Command “Get File Size” (command value 1Dh)

This command shall cause the File Server to respond the size of a file specified by the path name. If the path name leads to a directory, the size shall be returned as 0.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14
	high	mhigh	mlow	low
Return Code	File Size (32 bit unsigned integer)			

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Get File Size”. The File Server shall respond the file size always as a 32 bit unsigned integer as defined above. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server. The file size is not transferred.
22h	Not found	The current path name does not lead to any known file or directory. The file size is not transferred.
24h	Access denied	The File Server shall respond with this Return Code if the file size cannot be evaluated. The file size is not transferred. NOTE 20 This may be because the file is currently internally written, because another File Client is currently writing into this file, etc. This is not differentiated any further.

4.8.3.3.4.6 Command “Get Empty Disk Space” (command value 1Eh)

This command shall cause the File Server to respond the remaining disk space of the File Server.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10
Command Value

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14
	high	mhigh	mlow	low
Return Code	Empty Disk Space (32 bit unsigned integer)			

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Get Empty Disk Space”. The File Server shall respond the empty disk space always as a 32 bit unsigned integer as defined above. If the empty disk space is larger than 4’294’967’295 (FFFFFFFFh) octets, then also this maximum number is returned. It has to be interpreted as 4’294’967’295 octets **or more**. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server. The Empty Disk Space number is not transferred.

4.8.3.3.4.7 Command “Abort” (command value 1Fh)

This command shall cause the File Server to stop a running file transfer that has been initiated by any of the commands

- Retrieve File,
- List Directory or
- Get File.

NOTE 21 If a File Client wants to stop a post or store command, it signals the “end of transmission” by using a A_FileStream_InfoReport-PDU with a data length of 0.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10
Command Value

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Abort”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”.
10h	Invalid command	The given command is reserved or not supported by this File Server.
21h	Command not possible	The File Server shall respond with this Return Code if there is no ongoing file transfer to this File Client.

4.8.3.3.5 Command “Get File” (command value 21h)

This command shall cause the File Server to transfer a copy of a file specified in the path name, to the client. During the transfer, the file may be treated by a pre- or post processor like PHP. A content type parameter is returned to the client, which defines the content of the file in more detail. With this information, the client can treat upcoming files in different ways (see also [07] “Short HTML File Format”). The status and contents of the file in the server shall be unaffected. For the transmission of the file the Management Procedure HTTP_GetFile shall be applied as specified in [08] clause “File Transfer Procedures”, this is, the service A_FileStream_InfoReport shall be called consecutively by the File Server.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet n
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12
	Media-Type	Subtype
Return Code	Content-Type	

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command “Retrieve File”. The File Server shall handle this request as specified in [08] clause “File Transfer Procedures”. The returned **Content-Type** is described below. It may easily be converted for http use.
10h	Invalid command	The given command is reserved or not supported by this File Server. The returned **Content-Type** is not transferred.
22h	Not found	The current path name does not lead to any known file. The returned **Content-Type** is not transferred.
24h	Access denied	The File Server shall respond with this Return Code if the file cannot be read. The returned **Content-Type** is not transferred. NOTE 22 This may be because the file is currently internally written, because another File Client is currently writing into this file, etc. This is not differentiated any further.

Possible Content-Types

Media-Type	Subtype	Content-Type Description
00h	00h	text/plain
	01h	text/html
	02h	text/short-html-200 (see also “Short HTML File Format in [07])
	03h	text/short-html-320 (see also “Short HTML File Format in [07])
	04h	text/short-html-480 ((see also “Short HTML File Format in [07])
	05h	text/short-html-640 (see also “Short HTML File Format in [07])
	06h	text/short-html-800 (see also “Short HTML File Format in [07])
	07h	text/short-html-1024 (see also “Short HTML File Format in [07])
	08h to FFh	reserved
01h	00h	image/jpeg
	01h	image/gif
	02h	image/png
	03h	image/tiff
	04h	image/g3fax
	05h to FFh	reserved
02h	00h	audio/basic
	01h to FFh	reserved
03h	00h	video/mpeg
	01h to FFh	reserved

Media-Type	Subtype	Content-Type Description
04h	00h	application/octet-stream
	01h	application/postscript
	02h	application/pdf
	03h	application/zip
	04h to FFh	reserved
05h to FFh	00h to FFh	reserved

NOTE 23 The Content Type definition is taken from the MIME-Type (Multipurpose Internet Mail Extension) definition described in RFC 2045, RFC 2046 and others.

4.8.3.3.5.1 Command "Post File" (command value 22h)

This command shall cause the File Server to accept data from the File Client and store them as a file in its file system. If the file exists, then its contents shall be replaced. A new file shall be created if the file specified in the path name does not already exist. If the new file name is longer than the acceptable file name length, the file system may adjust the new file name automatically. For the transmission of the file the Management Procedure HTTP_PostFile shall be applied as specified in [08] clause "File Transfer Procedures", this is, the service A_FileStream_InfoReport shall be called consecutively by the File Client.

Property File Command Write (A_FunctionPropertyCommand-PDU)

octet 10	octet 11	octet 12 … octet N
Command Value	File Path	

or

octet 10
Command Value

If the command is given without a File Path, the Property PID_FILE_PATH shall be used instead and therefore has to be set before the command is executed. If a File Path is included in the command, the Property PID_FILE_PATH shall be overridden.

Property File Command Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

Possible Command Results

Return Code	Code Name	Description
00h	Command successful	This Return Code shall be used by the File Server to indicate that it accepts the command "Store File". The File Server shall handle this request as specified in [08] clause "File Transfer Procedures".
10h	Invalid command	The given command is reserved or not supported by this File Server.
22h	Not found	The current path name does not lead to any known directory.
23h	Write protected	The specified file is write protected.

Return Code	Code Name	Description
24h	Access denied	The File Server shall respond with this Return Code if the file cannot be stored. NOTE 24 This may be because the file is currently internally written, because another File Client is currently writing to the file, etc. This is not differentiated any further.
25h	No more space available	The File Servers disk or storage memory is full. Incoming data cannot be stored.

4.8.3.4 Usage by the Management Server

The Management Server shall handle the Property File Command as specified for the separate commands above.

Invalid (Reserved) and not supported commands shall be responded as specified in Table 27.

4.8.3.5 Usage by the Management Client (FTP client)

If a “Retrieve File”, “List Directory” and “Get File” command has a response with no error to the client and no data are transferred, the client shall detect this “error” after a certain time-out.

NOTE 25 This time-out may depend on the implementation of the File Server, the size and complexity of the file system and on the command itself, and may last up to 30 s and more. No generally applicable time-out values can be given.

4.8.4 Property File Path (PID = 53)

4.8.4.1 Format

Property value index				
0 (start-index)	1	2	…	N
Length of FilePath	File Path			
	octet 1	octet 2	…	octet N

Figure 6 – Property File Path

This Property shall be mandatory and shall contain the path name in absolute notation i.e always starting from the root directory e.g. „/dirname/filename.txt“.

The maximum length of this Character Array depends on the implementation. This maximal length can be retrieved by the File Client by reading the Property Description with an A_PropertyDescription_Read and interpreting the field max_nr_of_elem.

File Servers that handle only one file may have this Property implemented as “read only”.

The used file system shall be compatible to UNIX / LINUX. The filepath shall be encoded according DPT_UTF-8 (DPT_ID = 28.001). The Property Datatype shall be PDT_UTF-8.

A path name shall at least be coded in standard 7 bit (MSB = 0) ASCII code using only uppercase letters, lowercase letters, numbers and the following four special characters: “/” (slash), “-“ (hyphen), “_” (underline) and “.” (dot). 7 bit ASCII code is a subset of UTF-8 code.

Wildcards like “*.txt” shall not be used. Files shall be retrieved, renamed, deleted and so on, one after the other.

4.8.4.2 Usage by the Management Server (device)

There are no system variables like “current working directory“.

This is due to the fact that there is no open and close command to initiate and leave a command session (where system variables could be initialized). This requires that the path name is always in absolute notation.

The path name shall be deleted after the execution of any command. This rule only applies if the Property is “read / write”. A “read only” Property cannot be deleted.

4.8.4.3 Usage by the Management Client (FTP client)

The File Client shall set the Property PID_FILE_PATH to the value of the path name to which the subsequent command shall apply. If the path name is part of the command datagram, then this path name is used instead of the Property PID_FILE_PATH i.e. the Property PID_FILE_PATH shall be overridden.

4.8.5 Property MAX. APDU-Length (PID = 56)

This Property shall be an optional read-only Property and shall provide the maximum APDU-Length that is supported by this device for the A_FileStream_InfoReport-service. If this Property is not present the maximum APDU-Length of the device is valid for the A_FileStream_InfoReport-service (this is the value of the Property PID_MAX_APDU_LENGTH in the Device Object, or the value 14 if this Property is not available).

4.9 RF Medium Object (Object Type 19)

4.9.1 Overview

The RF Medium Object shall deal with the features of the KNX RF Communication Medium like:

- physical acknowledge, and
- Slow/Fast frequency transmission/reception, and
- Call RF channel.

This Interface Object shall be used to get the physical requirements of the RF device and shall thus enable the MaC to verify the compatibility between KNX RF devices in terms of RF capabilities (e.g. RF scanning and RF transmission). According to the physical requirements of each device, the MaC may enable or not the link between devices. The MaC shall be able to give additional information to the device along with data of the link (e.g. Fast or Slow transmission, acknowledge slot number…).

4.9.2 RF Medium Object for cEMI

Some of the cEMI Additional Information for KNX RF Multi may also be available in the local management between cEMI Client and cEMI Server.

The local management gives the possibility to a manufacturer to define dedicated behaviour of the cEMI Server device, independently of any L_Data messages.

EXAMPLE 6 A script that contains KNX RF Multi data Frames can be run in different modes. The same list of RF Frames is re-used, several times, just preceded by a different setting of the Management Server.

Some of the relevant local management Properties are placed within the RF Medium Object. Please refer to [15] for the conditions on the mandatory - or optional support of the RF Medium Object and the Properties.

These Properties are used mainly to configure the cEMI Server’s Physical Layer. They shall not be used to cause the transmission of an RF message.

These are some examples for local device management functions.

- Setting the reception mode of the cEMI Server (e.g. Fast, Slow, Fast & Slow)
- Transmitting a carrier on a specific frequency.
- Transmitting a modulated signal on a specific frequency.
- ….

For cEMI Server local management, the RF Medium Object (Object Type 10) may hold the Properties as specified in Table 30.

Table 30 – Properties in the RF Medium Object (informative)

Property Identifier	PID	Description, remark
PID_RF_MULTI_TYPE	51	Set the KNX RF Type to ready or to Multi.
PID_RF_MULTI_PHYSICAL_FEATURES	52	Expose the supported features of KNX RF Multi.
PID_RF_MULTI_CALL_CHANNEL	53	Set the Call Channel for KNX RF Multi Fast - and Slow frequencies.
PID_RF_MULTI_OBJECT_LINK	54	GA management of the KNXRF Multi specific features.
PID_RF_MULTI_EXT_GA_REPEATED	55	Managament of repeated KNX RF Multi GAs.
PID_TRANSMISSION_MODE	70	To manage the KNX RF Multi transmission mode of the RF Physical Layer
PID_RECEPTION_MODE	71	To manage the KNX RF Multi reception mode of the RF Physical Layer
PID_TEST_SIGNAL	72	To manage the transmission of a test signal on the specific frequency
PID_FAST_ACK	73	To manage the Fast Ack in reception
PID_FAST_ACK_ACTIVATE	74	To activate or deactivate the Fast Ack management in reception

4.9.3 PID_RF_MULTI_TYPE (PID = 51)

4.9.3.1 Format

- Property name: RF Multi Type
- Property Datatype: PDT_BITSET8
- Datapoint Type: none

Field	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Name	reserved	reserved	reserved	reserved	reserved	reserved	reserved	RF-Multi Typ e
Values	0	0	0	0	0	0	0	0, 1

Bit	Name	Description	Encoding
0	RF-Multi Type	*RF-Multi Type* shall indicate whether the KNX RF device is configured as “KNX RF Ready” type or as “KNX RF Multi” type	0: KNX RF Multi Device set in KNX RF Ready type 1: KNX RF Multi Device set in KNX RF Multi type
1 to 7	reserved	These bits are reserved and shall always be 0.	not applicable

4.9.3.2 Usage by the Management Client

- If the parameter PID_RF_MULTI_TYPE is set to KNX RF Ready type value then it is recommended to use the Property PID_OBJECT_LINK in the Device Object for the link management.
- If the parameter PID_RF_MULTI_TYPE is set to KNX RF Multi type value then it is recommended to use the Property PID_RF_MULTI_OBJECT_LINK in the RF Medium Object for the link management.

The MaC shall use this Property to set the value of the KNX RF type in the device.

The MaC should not change this Property if it has already programmed links for this device. This avoids inconsistency in the device configuration (e.g. mixture of links added in RF-multi type mode with fast ack management and links added in RF-ready type mode without fast ack management).

4.9.3.3 Usage by the Management Server

This parameter shall only be used by KNX RF Multi devices.

This Property shall only be implemented in the KNX RF Multi stack. A native KNX RF 1.1 or KNX RF Ready Device shall not implement this Property and react according the Property error handling if this Property is addressed. This shall indicate that the device is a KNX RF 1.1 or native KNX RF Ready device.

- If this parameter is set to the KNX RF Ready type, then the KNX RF Multi device shall act as a KNX RF Ready device. The specific RF Multi functionalities like multi frequencies, fast physical acknowledge shall not be supported by the device. This value shall enable the KNX RF Multi device (set in KNX RF Ready type) to be compatible and configurable with KNX RF 1.1 devices.
- If this parameter is set to the KNX RF Multi type, then the KNX RF Multi device shall act as a KNX RF Multi device. It shall then be possible to manage all the RF Multi functionnalities of the device, like multi frequencies, fast physical acknowledge (according to the physical requirements of the device). If this parameter is set in the KNX RF Multi type, the device shall neither be compatible nor configurable with a KNX RF 1.1 device.

If this Property is read or written, then the response shall contain the KNX RF type value that is actually written in the device.

The device that answers with b_0= 0b shall be a KNX RF Multi device that is set previously in the “KNX RF Ready” type.

4.9.4 PID_RF_MULTI_PHYSICAL_FEATURES (PID = 52)

4.9.4.1 Format

• Property name:	RF Multi Physical Features
• Property Datatype:	PDT_BITSET8
• Datapoint Type:	none

Field	**b_7**	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
Name	reserved	reserved	reserved	PhysAck	TxSx	TxFx	Scan Sx	Scan Fx
Values	0	0	0	0, 1	0, 1	0, 1	0, 1	0, 1

Bit	Name	Description	Encoding
0	Scan Fx	scan of the fast Fx RF channels	0: The device does not scan the fast Fx RF channels in runtime and thus cannot receive Frames transmitted on a fast Fx RF channel. 1: The device scans the fast Fx RF channels in runtime and thus can receive Frames transmitted on a fast Fx RF channel.
1	Scan Sx	scan of the slow Sx RF channels	0: The device does not scan the slow Sx RF channels in runtime and thus cannot receive Frames transmitted on a slow Sx RF channel. 1: The device scans the slow Sx RF channels in runtime and thus can receive Frames transmitted on a slow Sx RF channel.
2	TxFx	transmission on the fast Fx RF channels	0: The device cannot transmit Frames on the fast Fx RF channels. 1: The device is able to transmit Frames on the fast Fx RF channels.
3	TxSx	transmission on the slow Sx channels	0: The device cannot transmit Frames on the slow Sx RF channels. 1: The device is able to transmit Frames on the slow Sx RF channels.
4	PhysAck	management (Tx and Rx)of the fast physical acknowledge	0: The device is not able to manage a fast physical acknowledge in runtime. 1: The device is able to manage a fast physical acknowledge.
5 to 7	reserved	These bits are reserved and shall always be 0.	not applicable

4.9.4.2 Usage by the Management Server (device)

This Property shall be read-only.

If this Property is read, then the Management Server shall respond with the RF Multi physical features value.

4.9.4.3 Usage by the Management Client

The MaC shall read the Property RF Multi Physical Features to learn the RF multi physical capabilities of the device.

4.9.5 PID_RF_MULTI_CALL_CHANNEL (PID = 53)

4.9.5.1 Format

• Property name:	RF Multi Call Channel
• Property Datatype:	PDT_GENERIC_01
• Datapoint Type:	none

Field	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Name	reserved	reserved	reserved	reserved	Fast Frequency Call Channel		Slow Frequency Call Channel	
Values	0	0	0	0				

Bit	Name	Description	Encoding
0 to 1	Slow Frequency Call channel	This field shall indicate the slow frequency call channel.	00b: Call channel is the S1 slow frequency. 01b: Call channel is the S2 slow frequency. 10b: reserved for future use. 11b: reserved for future use.
2 to 3	Fast Frequency Call channel	This field shall indicate the fast frequency call channel.	00b: Call channel is the F1 fast frequency. 01b: Call channel is the F2 fast frequency. 10b: Call channel is the F3 fast frequency. 11b: reserved for future use.
4 to 7	reserved	These bits are reserved and shall always be 0.	not applicable.

4.9.5.2 Usage by the Management Server

The KNX RF Multi devices shall use the Slow – and Fast RF call channels as given by this Property value.

If this Property is read or written, then the response shall contain the KNX RF call channel value that is actually written in the device.

4.9.5.3 Usage by the Management Client

The MaC shall use the Property *RF Multi Call Channel* to set the value of the RF Call channel for the slow and the fast frequencies of the device.

4.9.6 PID_RF_MULTI_OBJECT_LINK (PID = 54)

4.9.6.1 Format

• Property name:	RF Multi Object Link
• Property Datatype:	PDT_Function
• Datapoint Type:	none

4.9.6.1.1Write (A_FunctionPropertyCommand-PDU)

4.9.6.1.1.1 Format

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21
Flags	00	SN high	SN	SN	SN	SN	SN low	GA high	GA low	Handle high	Handle low

octet 22	octet 23
Physical	AckSlot Nb

Octet 12 to 17: Serial number

In case of RF Multi link added/deleted in sending mode and physical acknowledge requested in runtime, the field SN is the KNX Serial Number of the receiver.

In case of RF Multi link added/deleted in non-sending mode, the field SN is the KNX Serial Number part of the Extended Group Address.

Octet 10: Flags

The octet 10 shall indicate whether a link to the referred Group Object shall be established or shall be broken. In case the link shall be added, it shall also indicate whether the link shall be a sending Group Address or not.

Field	**b_7**	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
Name	reserved	reserved	reserved	reserved	reserved	reserved	Add/Delete	Sending
Values	0	0	0	0	0	0		

Bit	**Name**	**Description and Encoding**
0	Sending	In case the contained Extended GA is added, then this field shall indicate whether it shall be the sending GA for the referred GO or not. This field shall only be interpreted in case the flag "*Add/Delete*" equals 0; in case the field "*Add/Delete*" equals 1, the value of the flag *Sending* shall be don't care. 0: not sending: The contained GA shall not be the sending GA for the referred GO. 1: sending: The contained GA shall be the sending GA for the referred GO. **Error and exception handling** - If a link is added to a GO to which the contained GA is already linked, but different value of the field *Sending*, then this new value of the flag *Sending* shall be used. - A GA set as "sending" shall take precedence on other previous sending GA for that GO.
1	Add/Delete	It shall indicate whether the contained extended GA shall be added to or removed from the list of GA assigned to the referred GO. 0: add: The contained GA shall be added to the list of GAs assigned to the referred GO. 1: delete: The contained GA shall be removed from the list of GAs assigned to the referred GO
2 to 7	reserved	These bits are reserved and shall always be 0.

Octet 20 and 21: Group Object Handle

The octet 20 and 21 shall contain the high and low octets of the Group Object Handle and shall identify the Group Object on which the extended Group Address must be set or deleted.

Octet 22: Physical Features (Fx and Sx frequencies)

The octet 22 and 23 contain data to manage the KNX RF Multi links.

Field	**b_7**	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
Name	reserved	reserved	reserved	reserved	reserved			
Values	0	0	0	0	0			

Bit	Name	Description	Encoding
0		KNX RF Multi Fast Fx frequency bit:	• For an **Output** Datapoint 0: A Frame shall not be transmitted on a fast Fx frequency RF channel in runtime. 1: A Frame shall be transmitted on a fast Fx frequency RF channel in runtime. • For an **Input** Datapoint 0: No Frame is received on a fast Fx frequency RF channel in runtime. 1: A Frame may be received on a fast Fx frequency RF channel in runtime.
1		KNX RF Multi Slow Sx frequency bit:	• For an **Output** Datapoint 0: A Frame shall not be transmitted on a slow Sx frequency RF channel in runtime. 1: A Frame shall be transmitted on a slow Sx frequency RF channel in runtime. • For an **Input** Datapoint 0: No Frame is received on a slow Sx frequency RF channel in runtime. 1: A Frame may be received on a slow Sx frequency RF channel in runtime.
2		KNX RF Ready Fast F1 frequency bit	• For an **Output** Datapoint: 0: A Frame (KNX RF Ready Frame) shall not be transmitted on the fast F1 frequency RF channel in runtime. 1: A Frame (KNX RF Ready Frame) shall be transmitted on the fast F1 frequency RF channel in runtime • For an **Input** Datapoint: 0: No Frame (KNX RF Ready Frame) is received on a fast F1 frequency RF channel in runtime. 1: A Frame (KNX RF Ready Frame) may be received on a fast F1 frequency RF channel in runtime.
3 to 7	reserved	These bits are reserved and shall always be 0.	not applicable

Octet 23: Acknowledge Slot Number

The octets 22 and 23 shall contain data to manage the KNX RF Multi links.

Field	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Name	reserved	Physical Acknowledge	Physical Acknowledge Slot Number					
Values	0							

Bit	Name	Description	Encoding
0	Physical Acknow ledge	Physical Acknowledge Slot Number	• For an **output** Datapoint 00h to 3Fh: If a Frame is transmitted on this Datapoint, a physical acknowledge shall be expected to be received on the specified ack slot number in runtime for this link. This means that if the output Datapoint is linked to N receivers, then N links have to be added on this output Datapoint. Each link containing the KNX Serial Number and the ack slot number used by the receiver for the sending of the physical acknowledge. • For an **input** Datapoint 00h to 3Fh: If a Frame is received on this Datapoint, a physical acknowledge shall be sent in the specified ack slot number in runtime for this link.
6	Physical Acknow ledge		• For an **output** Datapoint 0: The Frame shall be transmitted with no physical acknowledge request in runtime. 1: The Frame shall be transmitted with a physical acknowledge request in runtime. • For an i**nput** Datapoint 0: If a Frame is received on this link, then the device shall never answer with a fast physical acknowledge because the sender will not request a physical acknowledge in runtime. The value in the physical slot number field shall not be used and shall be set to 0. 1: If a Frame is received on this link with the physical acknowledge bit set in the Frame, then the device shall answer with a fast physical acknowledge. The value of the acknowledge slot time shall be contained in the physical slot number field.
7	reserved	This bit is reserved and shall always be 0.	not applicable

4.9.6.1.1.2 Response (A_FunctionPropertyState_Response-PDU)

octet 10
return Code

The response shall contain the return code of the operation.

Return Code:

00h:	SUCCESS (assignment /deletion done or already existed / not present)
FDh:	ERR_RF_PHYSICAL (any element or combination of elements from the physical features is not supported or incorrect)
FEh:	ERR_OBJECT (object does not exist)
FFh:	ERR_TABLEFULL (no more space in link table)

4.9.6.1.1.2.1 Usage by the Management Client

The MaC shall use the function “RF Multi Object Link” to assign a new receiving GA to a GO or to delete a GA from a GO. Input parameters shall be the GO-Handle to identify the GO, the Extended Group Address to be assigned/deleted, physical features of the link (frequency and acknowledge slot number).

NOTE 26 The Extended Group Address may include the own KNX Serial Number of the device, in this case an internal assignment shall be established / deleted.

4.9.6.1.2 Read (A_FunctionPropertyState_Read-PDU)

4.9.6.1.2.1 Format

octet 10	octet 11
Code	Iterator

Octet 10: Code

The octet 10 shall indicate whether the read command is used to go through the list of links or to know which part of the link table is available.

Code:	00h:	Read the RF multi links through an iterator.
	01h:	Read unused and total size of the link table. In this case, the response shall contain the return code together with two octets for the number of free RF Multi links and two other octets of the total size of the RF Multi link table.

4.9.6.1.2.2 Response in case of Code = 00h (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21
Return code	Flags	SN high	SN	SN	SN	SN	SN low	GA high	GA low	Handle high	Handle low

octet 22	octet 23
Physical	AckSlotNb

Octet 11: Flags

The octet 11 shall indicate whether the contained GA is the sending GA for the referred GO or not.

Field	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Name	reserved	reserved	reserved	reserved	reserved	reserved	reserved	Sending
Values	0	0	0	0	0	0	0	

Bit	Name	Description	Encoding
0	Sending	This field *Sending* shall indicate whether the contained extended GA is the sending GA for the referred GO or not.	0: not sending: The contained GA shall not be the sending GAfor the referred GO. 1: sending: The contained GA shall be the sending GA for the referred GO.
1 to 7	reserved	These bits are reserved and shall always be 0.	not applicable

Octet 12 to 17: Serial number

In case of RF Multi link added/deleted in sending mode and physical acknowledge requested in runtime, the field SN is the KNX Serial Number of the receiver.

In case of RF Multi link added/deleted in non-sending mode, the field SN is the KNX Serial Number part of the Extended Group Address.

Octet 20 and 21: Group Object Handle

The octet 20 and 21 shall contain the high and low octets of the GO-Handle, to identify the GO on which the Extended Group Address shall be set or deleted.

Octet 22: Physical Features (Fx and Sx frequencies)

Field	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Name	reserved	reserved	reserved	reserved	reserved			
Values	0	0	0	0	0			

Bit	Name	Description	Encoding
0		KNX RF Multi FastFx frequency bit	• For an **output** Datapoint 0: A Frame shall not be transmitted on a fast Fx frequency RF channel in runtime. 1: A Frame shall be transmitted on a fast Fx frequency RF channel in runtime. • For an **input** Datapoint 0: No Frame shall be received on a fast Fx frequency RF channel in runtime. 1: A Frame may be received on a fast Fx frequency RF channel in runtime
1		KNX RF Multi Slow Sx frequency bit	• For an **output** Datapoint 0: A Frame shall not be transmitted on a slow Sx frequency RF channel in runtime. 1: A Frame shall be transmitted on a slow Sx frequency RF channel in runtime. • For an **input** Datapoint 0: No Frame shall be received on a slow Sx frequency RF channel in runtime. 1: A Frame may be received on a slow Sx frequency RF channel in runtime.
2		KNX RF Ready Fast F1 frequency	• For an **output** Datapoint 0: A Frame (KNX RF Ready Frame) shall not be transmitted on the fast F1 frequency RF channel in runtime. 1: A Frame (KNX RF Ready Frame) shall be transmitted on the fast F1 frequency RF channel in runtime. • For an **input** Datapoint 0: No Frame (KNX RF Ready Frame) shall be received on a fast F1 frequency RF channel in runtime. 1: A Frame (KNX RF Ready Frame) may be received on a fast F1 frequency RF channel in runtime.

Bit	Name	Description	Encoding
3 to 7	reserved	These bits are reserved and shall always be 0.	not applicable

Octet 23: Acknowledge Slot Number

Field	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Name	reserved	Physical Acknowledge	Physical Acknowledge Slot Number					
Values	0							

Bit	Name	Description	Encoding
5 to 0	Physical Acknowledge Slot Number	Physical Acknowledge Slot Number	• For an **output** Datapoint: 00h to 3Fh: If a Frame is transmitted on this Datapoint then a physical acknowledge shall be expected to be received on the specified ack slot number in runtime for this link. • For an **input** Datapoint: 00h to 3Fh: If a Frame is received on this Datapoint then a physical acknowledge shall be sent in the specified ack slot number in runtime for this link.
6	Physical Acknowledge	Physical Acknowledge	• For an **output** Datapoint: 0: The Frame shall be transmitted with no physical acknowledge request in runtime. 1: The Frame shall be transmitted with a physical acknowledge request in runtime. • For an **input** Datapoint: 0: If a Frame is received on this link, then the device shall never answer with a fast physical acknowledge. The value in the physical slot number field shall not be used and shall be set to 0. 1: If a Frame is received on this link with the physical acknowledge bit set in the Frame then the device shall answer with a fast physical acknowledge. The value of the acknowledge slot time shall be contained in the physical slot number field
7	reserved	This bit is reserved and shall always be 0.	not applicable

4.9.6.1.2.3 Usage by the Management Client

The MaC shall use the function Read “RF Multi Object Link” with Code = 00h to read all Extended Group Addresses assigned to the GOs in a device. Therefore this Property shall serve as an iterator over the Group Object Association Table. The input parameter shall be an iterator that shall count the links in the device.

If the Return Code equals to 00h, then the response shall contain the Extended Group Address, the associated Group Object handle, the value of the Flags octet and the physical features in case an additional Extended Group Address is available.

If no more links is present, the return code shall be FFh.

4.9.6.1.2.4 Response in case of Code = 01h (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14
Return code	NFL high	NFL low	TNL high	TNL low

Octet 11 to 12: NFL (Number of Free Links)

The octets 11 and 12 contain the high and low octets of the number of free links, in the link table.

Octet 13 to 14: TNL (Total Number of Links)

The octets 13 and 14 shall contain the high and low octets of the total number of links, in the link table.

4.9.6.1.2.5 Usage by the Management Client

The MaC shall use the function Read “RF Multi Object Link” with Code = 01h to read the extension capabilities of the device. By this, the MaC shall be able to know if it can add new links to the device, before adding the links.

If the Return Code equals to 00h, then the response shall contain the number of free (or unused) links and the total number of links.

If no more links is present, the return code shall be FFh.

4.9.7 PID_RF_MULTI_EXT_GA_REPEATED (PID = 55)

4.9.7.1 Format

• Property name:	RF Multi Extended Group Address Repeated
• Property Datatype:	PDT_FUNCTION
• Datapoint Type:	none

4.9.7.1.1 Write (A_FunctionPropertyCommand-PDU)

4.9.7.1.2 Format

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21
code	00	SN high	SN	SN	SN	SN	SN low	GA high	GA low	reserved	reserved

Octet 10: Code

Code:	00h:	set Repeated Extended Group Address
	01h:	delete Repeated Extended Group Address
	02h:	delete all entries of the array (octet from 12 to 21 are in this case not used)

Octet 12 to 19

This field shall contain the Extended Group Address to add/delete in the array.

This shall be an array of octets containing the Extended Group Addresses (KNX Serial Number on 6 octets and GA on 2 octets) and 2 reserved octets.

This array shall contain the Extended Group Addresses of the Frames that shall be repeated. All Frames that do not comply with any Extended Group Address in this array shall be discarded.

If the GA equals 0000h in the Extended Group Address then all Frames containing the KNX Serial Number of this Extended Group Address shall be repeated (whatever the GA is in the Frame).

In case of deletion of an entry from the table, the exact value of SN and GA (so the Extended Group Address) shall be removed from the table. If only the SN is given in the delete command, no further deletion is done on other extended Group Address entries that would contain the SN.

Octet 20 to 21

Reserved octets: these octets shall be set to 0.

4.9.7.1.3 Response (A_FunctionPropertyState_Response-PDU)

octet 10
Return Code

The response shall contain the return code of the operation.

Return Code

00h:	SUCCESS (assignment /deletion done or already existed / not present)
FFh:	ERR_TABLEFULL (no more space in repeated table)

4.9.7.2 Read (A_FunctionPropertyState_Read-PDU)

4.9.7.2.1 Format

octet 10	oc tet 11
C od e	Iterator

Octet 10: Code

The octet 10 shall indicate whether the read command is used to go through the list of repeated Group Addresses or to know which part of the repeated Group Address Table is available.

Code:	00h:	Read the RF Extended Group Zddress repeated through an iterator.
	01h:	Read unused and total size of the link table. In this, the response shall contain a return code together with two octets for the number of unused Extended Group Address repeated and two other octets of the total size of the Extended Group Address repetition table

4.9.7.2.2 Response in case of Code = 00h (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14	octet 15	octet 16	octet 17	octet 18	octet 19	octet 20	octet 21
Return code	Iterator	SN high	SN	SN	SN	SN	SN low	GA high	GA low	00	00

4.9.7.2.3 Usage by the Management Client

The MaC shall use the function Read “RF Multi Extended Group Address Repeated” with Code = 00h to read all the Extended Group Addresses that are repeated in a device. Therefore this Property shall serve as an iterator over the repeater table. The input parameter shall be an iterator that shall count the repeated Extended Group Addresses in the device.

If the ReturnCode equals 00h then the response shall repeat the iterator value from the Read and shall contain the return code 00h and the repeated Extended Group Address shall be available. If no more repeated Extended Group Address is present, the return code shall be FFh.

4.9.7.2.4 Response in case of Code = 01h (A_FunctionPropertyState_Response-PDU)

octet 10	octet 11	octet 12	octet 13	octet 14
Return code	NFT high	NFT low	TNT high	TNT low

Octet 11 to 12: NFT (Number of Free entries in the repetition Table)

The octets 11 and 12 contain the high and low octets of the number of free entries, in the repetition table.

Octet 13 to 14: TNT (Total Number of entries in the repetition Table)

The octets 13 and 14 shall contain the high - and low octets of the total number of entries, in the repetition table.

4.9.7.2.5 Usage by the Management Client

The MaC shall use the function Read “RF Multi extended group address repeated” with Code = 01h to read the extension capabilities of the device. By this, the MaC shall be able to know if it can add new entries to the repetition table in the device, before adding them.

If the Return Code equals to 00h, then the response shall contain the number of free (or unused) entries and the total number of entries in the repetition table.

If no free entry is present in the repetition table, the return code shall be FFh.

4.9.8 PID_TRANSMISSION_MODE ((PID = 70)

4.9.8.1 General requirements

• Property name:	Transmission Mode
• Property Datatype:	PDT_ENUM8
• Datapoint Type:	None

The data Property *Transmission Mode* shall control the transmission mode of the Physical Layer implemented in the cEMI Server.

The cEMI Client shall write the Property *Transmission Mode* to change the transmission mode of the Physical Layer implemented in the cEMI Server.

The cEMI Client shall read the Property *Transmission Mode* to get the current transmission mode of the cEMI Server’s Physical Layer.

The value of the Property PID_TRANSMISSION_MODE shall be formatted as an 8 bit enumeration data type and shall be interpreted as specified in Table 31.

The interpretation of PID_TRANSMISSION_MODE depends on the device in which the cEMI Server is hosted and is specified in the following clauses.

Table 31 - Interpretation of the data in PID_TRANSMISSION_MODE

Value	Description
00h	Reserved for future use of the KNX system
01h	Transmission on the single F1 frequency (RF1.R)
02h	Transmission on the single F1 frequency (RF1.M)
03h	Transmission on the single F2 frequency (RF1.M)
04h	Transmission on the single F3 frequency (RF1.M)
05h	Transmission on the single S1 frequency (RF1.M)
06h	Transmission on the single S2 frequency (RF1.M)
07h	Transmission on the fast Fx frequencies (RF1.M)
08h	Transmission on the slow Sx frequencies (RF1.M)
09h	Reserved for future use of the KNX system
0Ah	Transmission on the single F1 frequency (RF2.R)
0Bh	Transmission on the single F1 frequency (RF2.M)
0Ch	Transmission on the single F2 frequency (RF2.M)
0Dh	Transmission on the single F3 frequency (RF2.M)
0Eh	Transmission on the single S1 frequency (RF2.M)
0Fh	Transmission on the single S2 frequency (RF2.M)
10h	Transmission on the fast Fx frequencies (RF2.M)
11h	Transmission on the slow Sx frequencies (RF2.M)
12h	Reserved for future use of the KNX system
13h	Transmission on the single F1 frequency (RF5.M)
14h	Transmission on the single F2 frequency (RF5.M)
15h	Transmission on the single F3 frequency (RF5.M)
16h	Transmission on the single S1 frequency (RF5.M)
17h	Transmission on the single S2 frequency (RF5.M)
18h	Transmission on the fast Fx frequencies (RF5.M)
19h	Transmission on the slow Sx frequencies (RF5.M)
1Ah	Reserved for future use of the KNX system
1Bh to FFh	Reserved for future use of the KNX system

The cEMI Client may read the Property *Transmission Mode* to get the current transmission mode.

If a message is transmitted between the last setting of the transmission frequency to Sx or Fx (using writing the Property or AddInfo) and reading the Property, then the returned value shall not return value neither Fx nor Sx.

4.9.8.2 Intended use and mandatory behaviour

This Property is mainly intended for test - and debugging purposes.

The Property value should be stored in volatile memory. The Property value shall be cleared after a reset of the MaS or after disconnection from the Management Client (cEMI Client). After a reconnection of the MaC (cEMI Client), or after a reset, the default values should be used by the Management Server.

4.9.9 PID_RECEPTION_MODE (PID = 71)

4.9.9.1 General requirements

- Property name: Reception Mode
- Property Datatype: PDT_ENUM8
- Datapoint Type: None

The data Property *Reception Mode* shall control the reception mode of the Physical Layer implemented in the cEMI Server.

The cEMI Client shall write the Property *Reception Mode* to change the reception mode of the Physical Layer implemented in the cEMI Server.

The cEMI Client shall also read the Property *Reception Mode* to get the current reception mode of the cEMI Server's Physical Layer.

The value of the Property *Reception Mode* shall be formatted as an 8 bit enumeration data type and shall be interpreted as specified Table 32.

The interpretation of Property *Reception Mode* shall depend on the device in which the cEMI Server is hosted and is specified in the following clauses.

Table 32 - Interpretation of the data in PID_RECEPTION_MODE

Value	Description
00h	Reserved for future use of the KNX system
01h	Reception on the single F1 frequency (RF1.R)
02h	Reception on the single F1 frequency (RF1.M)[1]
03h	Reception on the single F2 frequency (RF1.M)
04h	Reception on the single F3 frequency (RF1.M)
05h	Reception on the single S1 frequency (RF1.M)
06h	Reception on the single S2 frequency (RF1.M)
07h	Reception on the fast Fx frequencies (RF1.M)
08h	Reception on the slow Sx frequencies (RF1.M)
09h	Reception on the fast Fx and the slow Sx frequencies (RF1.M)
0Ah	Reception on the single F1 frequency (RF2.R)
0Bh	Transmission on the single F1 frequency (RF2.M)
0Ch	Transmission on the single F2 frequency (RF2.M)
0Dh	Transmission on the single F3 frequency (RF2.M)
0Eh	Transmission on the single S1 frequency (RF2.M)
0Fh	Transmission on the single S2 frequency (RF2.M)
10h	Transmission on the fast Fx frequencies (RF2.M)
11h	Transmission on the slow Sx frequencies (RF2.M)
12h	Reception on the fast Fx and the slow Sx frequencies (RF2.M)
13h	Transmission on the single F1 frequency (RF5.M)

Value	Description
14h	Transmission on the single F2 frequency (RF5.M)
15h	Transmission on the single F3 frequency (RF5.M)
16h	Transmission on the single S1 frequency (RF5.M)
17h	Transmission on the single S2 frequency (RF5.M)
18h	Transmission on the fast Fx frequencies (RF5.M)
19h	Transmission on the slow Sx frequencies (RF5.M)
1Ah	Reception on the fast Fx and the slow Sx frequencies (RF5.M)
0Bh to FFh	Reserved for future use of the KNX system

NOTE 27 As the RF1.M and RF1.R physically use the same frequencies, both types of Frame can be received in this case.

4.9.9.2 Intended use and mandatory behaviour

This Property is mainly intended for test - and debugging purposes.

The Property value should be stored in volatile memory. The Property value shall be cleared after a reset of the MaS or after disconnection from the Management Client (cEMI Client). After a reconnection of the MaC (cEMI Client), or after a reset, the default values should be used by the Management Server.

4.9.10 PID_TEST_SIGNAL (PID = 72)

4.9.10.1 General requirements

• Property name:	Test Signal
• Property Datatype:	PDT_GENERIC_02
• Datapoint Type:	None

The Property *Test Signal* shall control the frequency and the form of the signal to be transmitted by the cEMI Server’s Physical Layer.

The cEMI Client shall write the Property *Test Signal* to request the transmission of a signal for debugging purposes.

The value of the Property *Test Signal* shall be formatted as two consecutives 8 bit enumeration and shall be interpreted as specified in Table 33.

The interpretation of the Property *Test Signal* shall depend on the device in which the cEMI Server is hosted and is specified in the following clauses.

Table 33 - Interpretation of the data in PID_TEST_SIGNAL

Field	Octet 1	Octet 2	Octet 3
Name	Signal frequency	Signal form	Signal duration

Octet	Name	Description	Encoding
1	Signal frequency	Indicate the frequency of the signal or stop the signal.	00h: stop the current carrier transmission and resumes reception 01h: F1 Ready frequency (RF1.R) 02h: F1 frequency (RF1.M) 03h: F2 frequency (RF1.M) 04h: F3 frequency (RF1.M) 05h: S1 frequency (RF1.M) 06h: S2 frequency (RF1.M) 07h: Reserved for future use of the KNX system 08h: Reserved for future use of the KNX system 09h: Reserved for future use of the KNX system 0Ah: F1 Ready frequency (RF2.M) 0Bh: F1 frequency (RF2.M) 0Ch: F2 frequency (RF2.M) 0Dh: F3 frequency (RF2.M) 0Eh: S1 frequency (RF2.M) 0Fh: S2 frequency (RF2.M) 10h: Reserved for future use of the KNX system 11h: Reserved for future use of the KNX system 12h: Reserved for future use of the KNX system 13h: F1 frequency (RF5.M) 14h: F2 frequency (RF5.M) 15h: F3 frequency (RF5.M) 16h: S1 frequency (RF5.M) 17h: S2 frequency (RF5.M) 18h: Reserved for future use of the KNX system 19h: Reserved for future use of the KNX system 1Ah: Reserved for future use of the KNX system 1Bh to FFh: Reserved for future use of the KNX system
2	Signal form	Indicate the form of the signal	00h: carrier on fc - fdev 01h: carrier on fc + fdev 02h: 0/1 chips modulated signal 03h to FFh: Reserved for future use of the KNX system
3	Signal duration	Indicate the duration of the signal	00h: no time duration (permanent transmission) 01h to FFh: duration in seconds

f_c : centre frequency.

f_{dev} : FSK deviation.

The cEMI Client shall read the Property *Test Signal* to get the current status of the carrier command. If the cEMI Server is transmitting a carrier, it shall return the used signal frequency and the signal form. If there is no current transmission of a signal, the octet 1 shall contain the value 00h and the Signal form octet shall have no meaning.

The new start of a test signal shall stop the possible current test signal and restart the transmission of the new test signal.

4.9.10.2 Intended use and mandatory behaviour

This Property is mainly intended for test - and debugging purposes.

The Property value should be stored in volatile memory. The Property value shall be cleared after a reset of the MaS or after disconnection from the Management Client (cEMI Client). After a reconnection of the MaC (cEMI Client), or after a reset, the default values should be used by the Management Server.

4.9.11 PID_FAST_ACK (PID = 73)

4.9.11.1 General requirements

• Property name:	Fast Ack
• Property Datatype:	PDT_GENERIC_02[]
• Datapoint Type:	none

The data Property *Fast Ack* shall control the Fast Ack Slots that the cEMI's Physical Layer has to manage when receiving a RF message.

The cEMI Client shall write the Property *Fast Ack* for each Fast Ack slot that the cEMI Server device's Physical Layer shall fill along with the Info Octet to write in the Fast Ack.

The value of the data Property *Fast Ack* shall be formatted as 2 octets as specified in Table 34.

The interpretation of the Property *Fast Ack* shall depend on the device in which the cEMI Server is hosted and is specified in the following clauses.

The Fast Ack shall be organised as an array of 64 elements. Each element shall be a structured data made of two octets as described in Table 34.

The array shall be composed of 64 elements. The first element (first Fast Ack at Ack Slot 0) shall be located at index 1 of the array. The last element (last Fast Ack at Ack Slot 63) shall be located at index 64.

Table 34 - Interpretation of the data in PID_FAST_ACK

Field	**Octet 1**	**Octet 2**
Name	**Enable/Disable**	**Fast Ack Info Octet**
Values		

Table 35 – PID_FAST_ACK property description

Octet	**Name**	**Description**	**Encoding**	
1	Enable/Disable	Indicate if the management of the Ack Slot should be enabled or disabled	00h: 01h: 02h to FFh:	Disable Enable Reserved for future use of the KNX system
2	Fast Ack Info byte	Indicate the Fast Ack Info byte value to write in the related Fast Ack.	0 to 255: value	Fast Ack Info byte

The cEMI Client shall read the Property *Fast Ack* to get one or several Fast Ack management status. This shall be done by using the NumberOfElement and StartIndex parameters. One element (Fast Ack) shall be composed of 2 octets and the 2 octets shall be the same as the 2 octets that are written for this Fast Ack in the write request.

Multiple Fast Ack elements can be read in the read request by setting the right NumberOfElement and StartIndex parameters.

4.9.11.2 Intended use and mandatory behaviour

This Property is mainly intended for test - and debugging purposes.

The Property value should be stored in volatile memory. The Property value shall be cleared after a reset of the MaS or after disconnection from the Management Client (cEMI Client). After a reconnection of the MaC (cEMI Client), or after a reset, the default values should be used by the Management Server.

4.9.12 PID_FAST_ACK_ACTIVATE (PID = 74)

4.9.12.1 General requirements

• Property name:	Fast Ack Activate
• Property Datatype:	PDT_BINARY_INFORMATION
• Datapoint Type:	None

The Property *Fast Ack Activate* shall control the activation and deactivation of the Fast Ack management by the cEMI Server's Physical Layer when receiving RF messages with Fast Ack management required.

The cEMI Client shall write the Property *Fast Ack Activate* with value 1 to activate the management of the Fast Ack by the cEMI Server.

The value of the Property *Fast Ack Activate* shall be formatted as an 8 bit field data type and shall be interpreted as specified in Table 36.

The interpretation of PID_FAST_ACK_ACTIVATE shall depend on the device in which the cEMI Server is hosted and is specified in the following clauses.

Table 36 - Interpretation of the data in PID_FAST_ACK_ACTIVATE

Field	b7	b6	b5	b4	b3	b2	b1	b0
Name	reserved	reserved	reserved	reserved	reserved	reserved	reserved	Activation deactivation
Values	0	0	0	0	0	0	0	

Table 37 – PID_FAST_ACK_ACTIVATE property description

Octet	Name	Description	Encoding
0	Activation/ deactivation	This field shall indicate if the Fast Ack management has to be activeted or deactivated in the cEMI Server	0b: deactivate the Fast Ack management 1b: activate the Fast Ack management
1 to 7	Reserved for future use of the KNX system	These bits are reserved and shall always be 0	Not applicable

The cEMI Client shall read the Property *Fast Ack Activate* to get the current state, activated or deactivated, of the Fast Ack management in the cEMI Server device's Physical Layer.

Intended use and mandatory behaviour

This Property is mainly intended for test - and debugging purposes.

The Property value should be stored in volatile memory. The Property value shall be cleared after a reset of the MaS or after disconnection from the Management Client (cEMI Client). After a reconnection of the MaC (cEMI Client), or after a reset, the default values should be used by the Management Server.

4.10 Group Address Table (GrAT)

4.10.1 Abstract Resource definition

The GrAT shall be a shared Resource of both the Data Link Layer and the group oriented Transport Layer.

The Data Link Layer shall use the Group Address Table as a look up reference to check whether it should pass a received frame to the upper layers or not. The group oriented Transported Layer shall consult the Group Address Table to map an incoming LSAP (Group Address) to a TSAP [9] in receiving direction and vice-versa in sending direction.

Default state (optional)

The default state is not standardised but is implementation specific. This may be any of the following. Other default states are possible as well.

- The Group Address Table contains no Group Addresses.
- The Group Address Table contains an implementation specific number of default Group Addresses; the size has an according value.

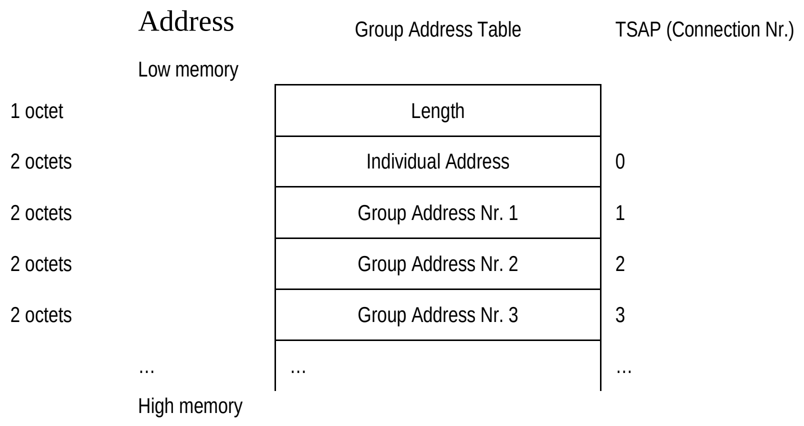
If the Group Address Realisation is controlled through a Load State Machine, then this may be Loaded or Unloaded.

4.10.2 Group Address Table - Realisation Type 1

Used by: Mask 0010h, 0011h, 0012h
Mask 0020h in compatibility mode , 0021h in compatibility mode

4.10.2.1 Format

Structure:



The Group Addresses shall be sorted in ascending order with increasing memory locations. Connection Nr. 1 refers to the first Group Address.

Max. Length: The maximum length is implementation dependent. It shall be checked whether this maximum value is not exceeded.

GAs: 16 bit

4.10.2.2 Location

The location shall be fixed at address 0116h.

[9] TSAP may also be referenced by "cr_id" ("communication reference identifier" or "communication relationship identifier"). Previous publications may use the term "connection number".

4.10.2.3 Usage by the Management Server (device)

4.10.2.3.1 Data Link Layer

The following specification is only valid for the L_Data Frame format. For the other frame formats (L_Busmon, L_Sys_Data), the Group Address Table shall not evaluated.

a) Frame transmission

The Group Address Table is not evaluated.

b) Frame reception

If Data Link Layer finds the frame it has composed from the Physical Layer to be correct, it shall evaluate the contained Domain Address (if relevant), destination_address and destination_address_flag (DAF) for deciding on passing to the frame to the Data Link Layer User or not.

The Data Link Layer shall only evaluate the GrAT under the following conditions:

1. the DoA equals the DoA of the device (if relevant), and
2. the DAF indicates the Destination Address is a Group Address, and
3. the frame contains a group message [10)].

If these conditions are not fulfilled, the behaviour shall be as described in the Data Link Layer specifications for the various media.

The GrAT shall be evaluated as follows:

♦ The behaviour of the Data Link Layer shall depend on the length of the Group Address Table (address 116h)

The Length (GroupAddressTableLength) shall contain the number of Group Addresses in the Group Address Table including the Individual Address.

EXAMPLE If there are 5 Group Addresses in the Group Address Table, the GroupAddressTableLength = 6.

Length	Behaviour
0	♦ Frame acceptance The Data Link Layer shall pass all group frames to the Data Link Layer User irrespective of the contained Group Address. ♦ Frame acknowledgement The condition for Group Frame acknowledgment based on the evaluation of the Group Address Table shall be considered as fulfilled [11)].
1	♦ Frame acceptance No Group Frame shall be passed to the Data Link Layer User. ♦ Frame acknowledgement No Group Frame shall be acknowledged.

[10] Thus no broadcast message.

[11]) This means that the acknowledge condition resulting from the Group Address Table evaluation is positive. Other conditions, typically medium specific, additionally apply for confirming the message.

Length	Behaviour
>1	The Data Link Layer shall evaluate the Group Address Table as follows. The Data Link Layer shall compare the destination_address to the subsequent Group Addresses in the Group Address Table, starting from connection nr. 1. This procedure shall stop on either one of the 3 following stop conditions: 1. the destination_address is found in the Group Address Table, or 2. the value of the destination_address is lower than the last compared Group Address of the Group Address Table [12]. 3. the last compared Group Address entry in the Group Address Table has the connection_number equal to the Address Table length, this is when the end of the Group Address Table is reached. ♦ Frame acceptance Only if the stop condition 1 is fulfilled, this is, if the destination_address is found in the Group Address Table, the Data Link Layer shall pass the received Group Frame to the Data Link Layer User by means of an L_Data-service. For the other stop conditions, the received Group Frame shall not be passed to the Data Link Layer User and the Group Frame shall be removed. ♦ Frame acknowledgement If the device does not support the non-selective Data Link Layer acknowledge, then only if the stop condition 1 is fulfilled, the condition for Group Frame acknowledgment based on the evaluation of the Group Address Table shall be regarded as fulfilled. If the device supports non-selective Data Link Layer acknowledge the condition for Group Frame acknowledgment based on the evaluation of the Group Address Table may be always fulfilled.

4.10.2.3.2 Transport Layer

The Transport Layer specifications (see [05]) specify that the TSAP used in the T_Data_Group service shall be a reference to a Group Address.

If using this standardized Group Address Table format, the TSAP shall be equal to the Connection Number.

a) Frame transmission

The Transport Layer shall use the value of the TSAP parameter in the T_Data_Group.req service primitive as connection number in the Group Address Table. The value of the Group Address found for this connection number shall be used as destination_address parameter in the N_Data_Group.req service primitive that shall be passed to the Network Layer.

b) Frame reception

The value of the parameter destination_address of the N_Data_Group.ind (N_Data_Group.con) shall be searched in the Group Address Table.

If the destination_address is found, the connection number at which it is found shall be taken as value for the parameter TSAP in the T_Data_Group.ind (T_Data_Group.con).

If the destination_address is not found, the value for the parameter TSAP in the T_Data_Group.ind shall be set to FFh to indicate an unknown connection number.

[12] For this reason, the Group Address entries in the Group Address Table have to be sorted. This allows for a fast evaluation algorithm.

4.10.2.4 Usage by the Management Client

4.10.2.4.1 Remote access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to download the Group Address Table Format 1.

It shall be checked on the forehand by the Management Server if the download of the Group Address Table does not conflict with other Resources [13]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;       Connect to the device
DMP_Connect_RCo
;       Set Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, 00h)
;       Set Group Address Table Length to 1 14)
DMP_MemWrite_RCo(0116h, 0116h, 01h)
;       Write the Group Address entries of the Group Address Table.
;       Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemWrite_RCo(0119h, 0119h+2*AddressTableLength-3, GrAT)
;       Set the Group Address Table length back to the value correct for the application.
;       AddressTableLength is the length of the group address table as defined in
                ;       clause 4.10.2.1 "Format" above.
DMP_MemWrite_RCo(0116h, 0116h, LLh)
;       Check the correct Group Address Table Length
DMP_MemRead_RCo(0116h,0116h)
;       If the received length is not equal to the programmed length, abort the Configuration Procedure.
;       Clear Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, FFh)
;       Restart the device
DMP_Restart
```

[13] Examples
1. Group Address that have been inserted compared to the Group Address Table currently running in the Management Server, which would make the Group Objects Association Table invalid.
2. Increased number of Group Address which require a larger address space than currently available in the Management Client, forcing other recourses, such as the Group Objects Association Table, to be shifted.

[14] This makes the Link Layer does no longer pass L_Data-frames containing group communication to the upper layers

4.10.2.4.2 Local access

```
;	Connect to the device
DMP_Connect_LEmi1
;	Set Runtime Error Flags
DMP_MemWrite_LEmi1(010Dh, 010Dh, 00h)
;	Set Group Address Table Length to 1
DMP_MemWrite_LEmi1(0117h, 0118h, PPPPh)
;	Write the Group Address entries of the Group Address Table.
;	Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemWrite_LEmi1 (0119h, 0119h+2*AddressTableLength-3, GrAT)
;	Set the Group Address Table length back to the value correct for the application.
;	AddressTableLength is the length of the group address table as defined in
		;	clause 4.10.2.1 "Format" above.
DMP_MemWrite_LEmi1 (0116h, 0116h, LLh)
;	Clear Runtime Error Flags
DMP_MemWrite_LEmi1 (010Dh, 010Dh, FFh)
;	Restart the device
DMP_Restart_LEmi1
```

4.10.2.4.3 Internal Access

There are no special conditions for an internal Management Client (application) to handle the Group Address Table – Realisation Type 1. The resources can be accessed directly either by the functionality provided by the host processor or by dedicated access routines (API) to handle the Resources.

In case this Resource is located in memory protected by a checksum, the internal Management Client shall take care of this, by either reducing the range of the checksum so that this Resource is not included, or by making this checksum is recalculated (for instance via a dedicated API).

4.10.3 Group Address Table - Realisation Type 2

Used by: Mask 0020h, 0021h

4.10.3.1 Format

The format is as specified in clause 4.10.2.1 “Format” above.

4.10.3.2 Location

The pointer to the GrAT shall be written to the Address Table Load State Machine in the Absolute Code/Data Allocation Record. For this location, the value of this pointer shall however not be evaluated. Instead, the Group Address Table shall be located at the fixed address 0116h.

4.10.3.3 Usage by the Management Server (device)

The usage by the Management Server for the Group Address Table Format 2 is as specified in clause 4.10.2.3 “Usage by the Management Server (device)” above.

4.10.4 Group Address Table - Realisation Type 4

Used by: Mask 1012h, 1013h

4.10.4.1 Format

Structure:

	Address	Group Address Table		TSAP (Connection Nr.)
	Low memory			
1 octet	0116h	Length		
2 octets	0117h	Individual Address		0
2 octets	0119h	R1	Group Address Nr. 1	1
2 octets	011Bh	R2	Group Address Nr. 2	2
2 octets	011Dh	R3	Group Address Nr. 3	3
	...	...	...	...
	High memory			

The Group Addresses shall be are sorted in ascending order with increasing memory locations. Connection Nr. 1 shall refer to the first Group Address.

Max. Length: The maximum length is implementation dependent. It shall be checked whether this maximum value is not exceeded.

GAs: 15 bit

The msb of every Group Address entry is the Group Responder Flag for that device for that Group Address (Rx = 1: the device is group responder for that Group Address [15]).

It shall be checked whether this maximum value is not exceeded.

4.10.4.2 Location

The location shall be fixed at address 0116h.

4.10.4.3 Usage by the Management Server (device)

4.10.4.3.1 Link Layer

The following specification is only valid for the L_Data Frame format. For the other frame formats (L_Busmon, L_Sys_Data), the Group Address Table shall not be evaluated.

The evaluation of the Group Address Table Realisation Type 4 shall be identical as specified in clause 4.10.2.3 "Usage by the Management Server (device)" above, with the additional evaluation of the bit Rx for every connection nr. X.

a) Frame transmission

The Group Address Table shall not be evaluated.

b) Frame reception

If Data Link Layer finds the frame it has composed from the Physical Layer to be correct, it shall evaluate the contained Domain Address (if relevant), destination_address and destination_address_flag (DAF) for deciding on passing the frame to the Data Link Layer User or not.

[15] Usage: See Powerline Communication Medium specification.

The Data Link Layer shall only evaluate the GrAT under the following conditions:

1. The DoA equals the DoA of the device (if relevant), and
2. t the DAF indicates the Destination Address is a Group Address, and
3. the frame contains a group message.

If these conditions are not fulfilled, the behaviour shall be as described in the Data Link Layer specifications for the various media.

The GrAT shall be evaluated as follows:

- The behaviour of the Data Link Layer shall depend on the length of the Group Address (0116h).

Length	**Behaviour**
0	♦ Frame acceptance The Data Link Layer shall pass all group frames to the Link Layer User irrespective of the contained Group Address. ♦ Frame acknowledgement The condition for Group Frame acknowledgment based on the evaluation of the Group Address Table shall be considered as fulfilled under the extra condition that the Main Responder Flag is set [16].
1	♦ Frame acceptance No Group Frame shall be passed to the Data Link Layer User. ♦ Frame acknowledgement No Group Frame shall be acknowledged.

[16] This means that the acknowledge condition resulting from the Group Address Table evaluation is positive. Other conditions, typically medium specific, additionally apply for confirming the message.

Length	Behaviour
>1	The Data Link Layer shall evaluate the Group Address Table as follows. The Data Link Layer shall compare the destination_address to the subsequent Group Addresses in the Group Address Table, starting from connection nr. 1. This procedure shall stop on either one of the 3 following stop conditions: 1. the destination_address is found in the Group Address Table, or 2. the value of the destination_address is higher than the last compared Group Address of the Group Address Table [17]. 3. the last compared Group Address entry in the Group Address Table has the connection_number equal to the Address Table length, this is when the end of the address table is reached. ♦ Frame acceptance Only if the stop condition 1 is fulfilled, this is, if the destination_address is found in the Group Address Table, the Data Link Layer shall pass the received Group Frame to the Data Link Layer User by means of an L_Data-service. For the other stop conditions, the received Group Frame shall not be passed to the Data Link Layer User and the Group Frame shall be removed. ♦ Frame acknowledgement The received frame shall be acknowledged only if both of the following conditions are fulfilled: 1. The Group Address Table evaluation procedure is stopped by the stop condition 1, this is, the Group Address is found in the Group Address Table, and 2. In the Connection X at which the Group Address is found, the bit Rx is set.

4.10.4.3.2 Transport Layer

In the Transport Layer specifications (see [05]) specify that the TSAP used in the T_Data_Group service shall be a reference to a Group Address.

If using this standardized Group Address Table format, the TSAP shall be equal to the Connection Number.

a) Frame transmission

The Transport Layer shall use the value of the TSAP parameter in the T_Data_Group.req service primitive as connection number in the Group Address Table. The value of the Group Address found for this connection number is used as destination_address parameter in the N_Data_Group.req service primitive which shall be passed to the Network Layer.

b) Frame reception

The value of the parameter destination_address of the N_Data_Group.ind (N_Data_Group.con) shall be searched in the Group Address Table.

If the destination_address is found, the connection number at which it is found shall be taken as value for the parameter TSAP in the T_Data_Group.ind (T_Data_Group.con).

If the destination_address is not found, the value for the parameter TSAP in the T_Data_Group.ind shall be set to FFh to indicate an unknown connection number.

[17] For this reason, the Group Address entries in the Group Address Table have to be sorted. This allows for a fast evaluation algorithm.

4.10.4.4 Usage by the Management Client

4.10.4.4.1 Remote access

The access by a remote Management Client is as specified in clause 4.10.2.4.1 "Remote access" above.

4.10.4.4.2 Local access

The access by a local Management Client is as specified in clause 4.10.2.4.2 "Local access" above.

4.10.4.4.3 Internal access

The access by an internal Management Client is as specified in clause 4.10.2.4.3 "Internal Access" above.

4.10.5 Group Address Table - Realisation Type 5

Used by: Mask 3012h, 4012h configured in S-Mode

The specification of this Resource is only valid if it is and will be managed in S-Mode configuration. If the device hosting this Resource is managed by another Configuration Mode, the locations for this Resource can still be accessed; however, these accessed values shall not be interpreted according the specifications below.

4.10.5.1 Format

Structure:

	Address	Group Address Table	Connection Nr. (TSAP)
	Low memory		
1 octet	0116h	Length	
2 octets	0117h	Individual Address	0
2 octets	0119h	Group Address Nr. 1	1
2 octets	011Bh	Group Address Nr. 2	2
2 octets	011Dh	Group Address Nr. 3	3
	…	…	…
	High memory		

The Group Addresses are not sorted. Connection Nr. 1 shall refer to the first Group Address.

Max. Length: The maximum length is implementation dependent. It shall be checked whether this maximum value is not exceeded.

GAs: 16 bit

4.10.5.2 Location

The location shall be fixed at address 0116h.

4.10.5.3 Usage by the Management Server (device)

4.10.5.3.1 Data Link Layer

The following specification is only valid for the L_Data Frame format. For the other frame formats (L_Busmon, L_Sys_Data), the Group Address Table shall not be evaluated.

a) Frame transmission

The Group Address Table shall not be evaluated.

b) Frame reception

If Data Link Layer finds the frame it has composed from the Physical Layer to be correct, it shall evaluate the contained Domain Address (if relevant), destination_address and destination_address_flag (DAF) and for deciding on passing to the frame to the Data Link Layer User or not.

The Data Link Layer shall only evaluate the GrAT under the following conditions:

1. The DoA equals the DoA of the device (if relevant), and
2. the DAF indicates the destination address is a group address, and
3. the frame contains a group message.

If these conditions are not fulfilled, the behaviour shall be as described in the Data Link Layer specifications for the various media.

The GrAT shall be evaluated as follows.

- ♦ The behaviour of the Data Link Layer shall depend on the length of the Group Address Table (address 0116h)

Length	**Behaviour**
0	♦ Frame acceptance The Data Link Layer shall pass all group frames to the Data Link Layer User irrespective of the contained Group Address. ♦ Frame acknowledgement when relevant The condition for Group Frame acknowledgment based on the evaluation of the Group Address Table shall be considered as fulfilled [18].
1	♦ Frame acceptance No Group Frame shall be passed to the Data Link Layer User. ♦ Frame acknowledgement No Group Frame shall be acknowledged.

[18] This means that the acknowledge condition resulting from the Group Address Table evaluation is positive. Other conditions, typically medium specific, additionally apply for confirming the message.

Length	Behaviour
>1	The Data Link Layer shall evaluate the Group Address Table as follows. The Data Link Layer shall compare the destination_address to the subsequent Group Addresses in the Group Address Table, starting from connection nr. 1. This procedure shall stop on either one of the 3 following stop conditions: 1. the destination_address is found in the Group Address Table, or 2. the last compared Group Address entry in the Group Address Table has the connection_number equal to the Address Table length, this is when the end of the address table is reached. ♦ Frame acceptance Only if the stop condition 1 is fulfilled, this is, if the destination_address is found in the Group Address Table, the Data Link Layer shall pass the received Group Frame to the Data Link Layer User by means of an L_Data-service. For the other stop condition, the received Group Frame shall not be passed to the Data Link Layer User and the Group Frame shall be removed. ♦ Frame acknowledgement Only if the stop condition 1 is fulfilled, the condition for Group Frame acknowledgment based on the evaluation of the Group Address Table shall be considered as fulfilled.

4.10.5.3.2 Transport Layer

The Transport Layer specifications (see [05]) require that the TSAP used in the T_Data_Group service shall be a reference to a Group Address.

If using this standardized Group Address Table format, the TSAP shall be equal to the Connection Number.

a) Frame transmission

The Transport Layer shall use the value of the TSAP parameter in the T_Data_Group.req service primitive as connection number in the Group Address Table. The value of the Group Address found for this connection number shall be used as destination_address parameter in the N_Data_Group.req service primitive that shall be passed to the Network Layer.

b) Frame reception

The value of the parameter destination_address of the N_Data_Group.ind (N_Data_Group.con) shall be searched in the Group Address Table.

If the destination_address is found, the connection number at which it is found shall be taken as value for the parameter TSAP in the T_Data_Group.ind (T_Data_Group.con).

If the destination_address is not found, the value for the parameter TSAP in the T_Data_Group.ind shall be set to FFh to indicate an unknown connection number.

4.10.5.4 Usage by the Management Client

4.10.5.4.1 Remote access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to download the Group Address Table - Realisation Type 5.

It shall be checked on the forehand by the Management Server if the download of the Group Address Table does not conflict with other resources [19]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;         Connect to the device
DMP_Connect_RCo
;         Set Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, 00h)
;         Set Group Address Table Length to 1 20)
DMP_MemWrite_RCo(0116h, 0116h, 01h)
;         Write the Group Address entries of the Group Address Table.
;         Note that this management procedure foresees the optional splitting up of the data
                    ;         in blocks of 12 octets.
DMP_MemWrite_RCo(0119h, 0119h+2*AddressTableLength-3, GrAT)
;         Set the Group Address Table length back to the value correct for the application.
DMP_MemWrite_RCo(0116h, 0116h, LLh)
;         Check the correct Group Address Table Length
DMP_MemRead_RCo(0116h,0116h)
;         If the received length is not equal to the programmed length, abort the Configuration Procedure.
;         Clear Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, FFh)
;         Restart the device
DMP_Restart
```

[19] Examples
1. Group Address that have been inserted compared to the Group Address Table currently running in the Management Server, which would make the Group Objects Association Table invalid.
2. Increased number of Group Address which require a larger address space than currently available in the Management Client, forcing other recourses, such as the Group Objects Association Table, to be shifted.

[20] This makes the Data Link Layer does no longer pass L_Data-frames containing group communication to the upper layers.

4.10.6 Group Address Table – Realisation Type 6

Used by: System 300

4.10.6.1 Location

The Group Address Table - Realisation Type 6 shall be implemented as Interface Object of the type Addresstable Object (Object Type = 0001h).

4.10.6.2 Format

The Address Table Interface Object shall only include the Group Addresses of the device. It shall provide the management operations for uploading and downloading of the Group Address Table.

The Group Address Table Realisation Type 6 shall support both the L_Data_Standard frames format as well as the L_Data_Extended frame format. For each Frame Format there shall be a corresponding Group Address Table. One AddressTable Object shall contain exactly one of the Properties Address Table Format 0 (PID_TABLE = 23) or Address Table Format 1 (PID_ADDRTAB1= 52) with one given Address Table format as specified below. A device may support multiple Frame Formats and therefore more than one instance of the Addresstable Object is possible. The Property "Address Table Format 1" shall only be valid for the frame format given in the Property "Extended Frame Format" (PID_EXT_FRAMEFORMAT = 51). If the Property "Extended Frame Format" does not exist the Frame Format shall be 0 (L_Data_Standard frame format).

The Group Address Table – Realisation Type 6 shall provide the Properties as specified in Table 42 and defined in detail in the further clauses. For the requirements about which Properties are mandatory and optional and for the access levels, please refer to Annex A in [15].

Table 38 – Group Address Table Interface Object – Realisation Type 6

Property Name	Property Identifier	PDT/DPT	Use
Interface Object Type	1 = PID_OBJECT_TYPE	See 4.2.1.	BCU/LTE
Interface Object Name	2 = PID_OBJECT_NAME	See 4.2.2.	BCU/LTE
Load Control	5 = PID_LOAD_STATE_CONTROL	See 4.2.5.	BCU/LTE
Address Table Format 0 [a]	23 = PID_TABLE	PDT_GENERIC_02[]	BCU/LTE
Extended Frame Format	51 = PID_EXT_FRAMEFORMAT	PDT_UNSIGNED_CHAR	LTE
Address Table Format 1 [a]	52 = PID_ADDRTAB1	PDT_GENERIC_04[]	LTE

[a] Each instance of the Group Address Table Object shall hold exactly one of the Properties "Address Table Format 0" or "Address Table Format 1".

4.10.6.2.1 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This Property shall contain Object Type of the Interface Object. The Group Address Table shall have Object Type 0001h.

4.10.6.2.2 PID_OBJECT_NAME (PID = 2)

Please refer to 4.2.2 for the general requirements for PID_OBJECT_NAME.

This Property shall contain the name of the Address Table.

4.10.6.2.3 PID_LOAD_STATE_CONTROL (PID = 5)

Please refer to 4.2.5 for the general requirements for PID_LOAD_STATE_CONTROL.

4.10.6.2.4 PID_TABLE (PID = 23)

This Property “Address Table Format 0” shall include an array with the Group Address Table. Only Group Addresses used by the device shall be in the table. If the current length is 0 all Group Addresses shall be accepted. No Group Address shall be accepted if the load control state differs from loaded (e.g.: unloaded or loading). The Group Addresses shall be filled in ascending order. The current length of the array shall be the number of entries in the Address Table. The maximum length shall be calculated via the Load Control “Relative Allocation” (for further information please refer to [08] clause “Load Control Record: Relative Allocation”) or shall be fixed in the device.

Table 39 – Property Address Table Format 0 - value

Property Value array index	**Description**	**Size**	TSAP
(current nr of elem) 0	Current length	2 octets	
1	Group Address 1	2 octets	1
2	Group Address 2	2 octets	2
…	…	…	
n	Group address n	2 octets	n

The maximum length of the Address Table Property shall be read with an A_PropertyDescription_Read service.

4.10.6.2.5 PID_EXT_FRAMEFORMAT (PID = 51)

This Property is reserved for LTE-Mode devices. The Extended Frame Format Property shall include the value of Extended Frame Format (0001b to 1111b) for which the Address Table Property shall be valid [21)]. If the Extended Frame Format Property does not exist the Address Table shall be valid for frame type 0 (standard frame).

4.10.6.2.6 PID_ADDRTAB1 (PID = 52)

This special LTE-Mode Property Address Table Format 1 shall control the support of Extended Frames. One address entry shall consist of four octets. The first two octets shall include the base address and the third and fourth a mask with the valid bits of the base address. The rule for address matching for reception shall be:

If (Group Address AND mask) is equal to (base address AND mask) then match

This Property shall only include the zone addresses for one Extended Frame Format. If more than one Extended Frame Format is supported in a device multiple instances of the Address Table Object with PID_ADDRTAB1 shall exist. The number of Address Table Objects is theoretically not limited in a device. The Address Table Format 1 Property shall only include values for reception, transmission addresses are specified in the application.

The maximum length shall be calculated via the Load Control “Relative Allocation” or shall be fixed in the device.

[21)] The Extended Frame Format field is specified in [03].

Table 40 - Address Table Format 1 - Example 1

Base Address	Mask	Matching addresses
1234h	FFFFh	1234h
1230h	FFFFh	1230h
1200h	FFF0h	1200h – 120Fh (sniffer)
1204h	FF0Fh	12x4h, x=0h - Fh (sniffer)
1534h	FFFFh	1534h
...		

Table 41 - Example for a device with more than one Address table

Address Table Instance	Interface Object Type	Local Index (device Object = 0)	Frame Format (PID = 51)	Address Table Property	Address Table Values	Description
1	1	1	00h	23	1001h, 1002h, 1003h,…	S-Mode Group Address Table
2	1	2	04h	52	1234.FFFFh, 1230.FFFFh, 1200.FFF0h 1204.FF0Fh 1534.FFFFh, 1530.FFFFh, 1500.FFF0h	Geographical Tag 1 Address Table
3	1	3	05h	52	2534.FFFFh, 2530.FFFFh, 2500.FFF0h	Geographical Tag 2 Address Table
4	1	4	06h	52	00A1.FFFFh	Application Specific Tag Address Table
5	1	5	07h	52	0654.FFFFh	Unassigned Tags Address Table

4.10.6.4 Usage by the Management Client

4.10.6.4.1 General

For the Property Based Management no memory allocation is needed. The data shall be written directly to the Property Address Table. The maximum size of the table can be read via a description read in the maximum number of elements in the Address Table Property. To prevent the connectionless access to Address Table it is recommended to not use the lowest access level for this Interface Object.

The complete download of the Group Address Table – Realisation Type 6 shall be done in the following way:

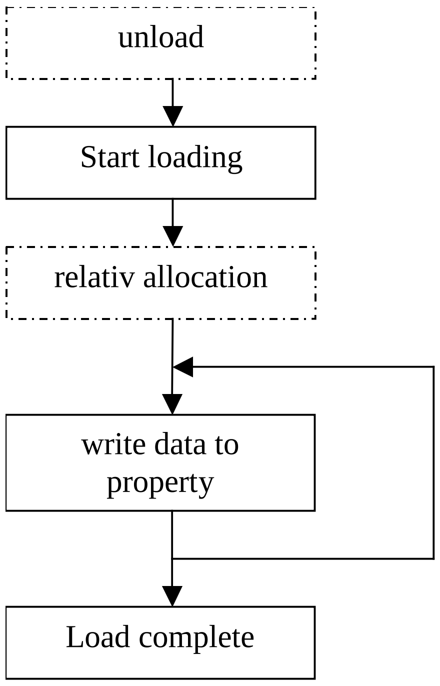


Figure 7 - Load Procedure

4.10.6.4.2 Remote Access

The following Load Controls shall be supported for the management of the Group Address Table – Realisation Type 6.

Load Control	Subtype	Description	Supported
00h		No operation	M
01h		Start Loading	M
02h		Load Completed	M
03h		Additional Load Controls	
	0Ah	Relative Allocation	M
04h		Unload	M

4.10.6.4.3 Local Access

Local Access by a Management Client to the Group Address Table – Realisation Type 6 is not specified.

4.10.7 Group Address Table – Realisation Type 7

Used by: System B

4.10.7.1 Location

The Group Address Table - Realisation Type 7 shall be implemented as Interface Object of the type Addresstable Object (Object Type = 0001h). This Interface Object shall provide the Property PID_TABLE_REFERENCE for memory mapped access to the table data.

4.10.7.2 Format

The Group Address Table Object shall contain the Group Address Table length, and the Group Addresses (LSAPs) of the device.

It shall provide the management operations for the up- and download of Group Address Tables.

The Group Address Table – Realisation Type 7 provides the Properties as specified in Table 42 and defined in detail in the further clauses. For the requirements about which Properties are mandatory and optional and for the access levels please refer to Annex A in [15].

Table 42 – Group Address Table Interface Object – Realisation Type 7

Property Name	Property Identifier	PDT/DPT
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL
Table Reference	7 = PID_TABLE_REFERENCE	PDT_UNSIGNED_LONG
Table	23= PID_TABLE	PDT_UNSIGNED_INT[]
Memory Control Table	27= PID_MCB_TABLE	PDT_GENERIC_08[]
Error code	28= PID_ERROR_CODE	PDT_ENUM8
PL110 Group Responser Table [a]	53= PID_GROUP_RESPONSER-_TABLE	PDT_BITSET8[]
[a] This Property is mandatory for PL110 devices. For all other media this Property shall not be implemented.		

4.10.7.2.1 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This Property shall be the Object Type of the Interface Object. The Group Address Table shall have Object Type 0001h.

4.10.7.2.2 PID_LOAD_STATE_CONTROL (PID = 5)

Please refer to 4.2.5 for the general requirements for PID_LOAD_STATE_CONTROL.

4.10.7.2.3 PID_TABLE_REFERENCE (PID = 07)

Please refer to 4.2.7 for the general requirements for PID_TABLE_REFERENCE.

4.10.7.2.4 PID_TABLE (PID = 23) for Group Address Table - Realisation Type 7

This Property shall contain the Group Address Table – Realisation Type 7.

The contents of this Property shall only be valid in the load state ‘loaded’. Writing shall only have effect in the load state ‘Loading’.

For the Realisation Type 7 of the Group Address Table, the Property Table shall contain an array with the Group Address Table. Only Group Addresses (LSAPs) used by the device shall be in the table. If the current length is 0 all Group Addresses shall be accepted. To accept no Group Address at all the Load State shall be set to a state different from “Loaded” (e.g.: “Unloaded” or “Loading”). The Group Addresses shall be filled in ascending order. The current length shall be the number of entries in the Address Table. The maximum length of the Address Table Property shall be read via A_Property-Description_Read. The Maximum Length shall be calculated via the load control *“relative allocation”* (for further information please refer to [08] clause “Load Event Data Relative Allocation” and [09] in the Configuration Procedures that use this Load Control) or shall be fixed in the device.

Table 43 - Group Address Table – Realisation Type 7

Property Value array index	Description	Size	TSAP
(current nr of elem) 0	Current length	2 octets	
1	Group Address 1	2 octets	1
2	Group Address 2	2 octets	2
…	…	…	
n	Group address n	2 octets	n

Every Group Address entry shall comply with the format and full range as specified in clause 3.4 and [03].

4.10.7.2.5 PID_GROUP_RESPONSER_TABLE (PID = 53)

This Property is mandatory for PL110 devices. For all other media this Property shall not be implemented.

Each bit in this bit field shall define the Data Link Layer acknowledge generation of one associated LSAP.

Table 44 - Group Responser Flag (GRF) Table

Octet#	Bit#							
	7	6	5	4	3	2	1	0
Octet 0	Max. Length (LSAPs DIV 8) + 1							
Octet 1	GRF 8	GRF 7	GRF 6	GRF 5	GRF 4	GRF 3	GRF 2	GRF 1
Octet 2	GRF 16	GRF 15	GRF 14	GRF 13	GRF 12	GRF 11	GRF 10	GRF 9
…								
Octet n	…	…	…	…	…	…	…	…

4.10.7.2.6 PID_MCB_TABLE (PID = 27)

Please refer to the Interface Object Type independent specification of PID_MCB_TABLE in 4.2.27.

For the Group Address Table Object only the checksum feature is intended to be used but using the same mechanism as for the Application Program objects.

For this case the maximal table length (no. of elements) shall be set to ‘1’ and the read and write access shall be set to the lowest access level.

4.10.7.2.7 PID_ERROR_CODE (PID = 28)

Please refer to 4.2.28 for the general requirements for PID_ERROR_CODE.

4.10.7.3 Usage by the Management Server

The usage by the Management Server (device) shall be identical as for Group Address Table – Realisation Type 4 (see 4.10.4), except for the frame acknowledgement.

♦ Frame acknowledgement

The received frame shall be acknowledged only if both of the following conditions are fulfilled:

1. The Group Address Table evaluation procedure is stopped by the stop condition 1, this is, the Group Address is found in the Group Address Table, and

2. In the Property PID_GROUP_RESPONSER_TABLE the bit is set at Property array element (Group Address DIV 8) + 1 at bit position (Group Address – 1) MOD 8.

4.10.8 Group Address Table – Realisation Type 8

Used by mask 5705h.

4.10.8.1.1 Format

Structure: Address

Low memory

	Group Address Table	TSAP (Connection Nr.)
1 octet	Length	
2 octets	Individual Address	0
2 octets	Group Address Nr. 1	1
2 octets	Group Address Nr. 2	2
2 octets	Group Address Nr. 3	3
	…	…

High memory

The Group Addresses shall be are sorted in ascending order with increasing memory locations. Connection Nr. 1 shall refer to the first Group Address.

Max. Length: The maximum length is implementation dependent. It shall be checked whether this maximum value is not exceeded.

GAs: 16 bit

4.10.8.2 Location

The Location shall be fixed at address 4000h.

4.10.9 Group Address Table - E-Mode Realisation Type 1 (GrAT – Easy 1)

Used by: Ctrl-Mode fixed DMA
and mask 0012h in E-Mode

4.10.9.1 Format

See clause 4.10.2.1.

4.10.9.2 Location

The location is fixed at address 0116h.

4.10.9.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and mask 0012h the following applies.

Before memory mapped access to a device, the E-Mode Controller shall set the error flags to 00h to stop the application. A device shall reject Link Services sending a negative response (start_index = 0 and no group_address_list) if error flags are set to 00h, not depending on communication mode (error flags definition is given in [18]).

An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.10.9.4 Usage by the Management Server (device)

See clause 4.10.2.3.

4.10.9.5 Usage by the Management Client

4.10.9.5.1 Remote access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to read and to write the GrAT – Easy 1 and to get the maximum length of the GrAT – Easy 1.

4.10.9.5.1.1 Read GrAT – Easy 1

```
;       Connect to the device
DMP_Connect_RCo
;       Get GrAT – Easy 1 Group Address Table Length
DMP_MemRead_RCo(0116h, 0116h, AddressTableLength)
;       Read the Group Address entries of the GrAT – Easy 1.
;       Note that this Management Procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemRead_RCo(0119h, 0119h+2*AddressTableLength-3, GrAT)
;       Disconnect from the device
DMP_Disonnect_RCo
```

4.10.9.5.1.2 Get maximum length GrAT – Easy 1

The calculation of the maximum length of the GrAT – Easy 1 is based on the Association Table following directly behind the GrAT – Easy 1 without space between the tables.

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Association Table Pointer
DMP_MemRead_RCo(0111h, 0111h, GrAssocTabPtr)
;       Disconnect from the device
DMP_Disonnect_RCo
;       Calculate the maximum length of GrAT – Easy 1:
;       ⇒      AddressTableMaxLength = (0100h + GrAssocTabPtr - 0116h - 3)/2 22)
```

4.10.9.5.1.3 Write GrAT – Easy 1

It shall be checked on beforehand by the Management Server if the download of the GrAT – Easy 1 does not conflict with other Resources [23]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;       Connect to the device
DMP_Connect_RCo
;       Set Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, 00h)
;       Set Group Address Table Length to 1 24)
```

[22] The calculation is based on the memory block reserved for GrAT – Easy 1, including 3 octets for GrAT – Easy 1 Table length and the Individual Address. The Association Table directly follows in memory the GrAT – Easy 1 (please refer to [09] "Appendix 5 : Controller Mode download Procedures").

[23] EXAMPLE Group Address that have been inserted compared to the GrAT – Easy 1 currently running in the Management Server, which would make the Group Objects Association Table invalid.

```
DMP_MemWrite_RCo(0116h, 0116h, 01h)
;         Write the Group Address entries of the Group Address Table.
;         Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemWrite_RCo(0119h, 0119h+2*AddressTableLength-3, GrAT)
;         Set the Group Address Table length back to the value correct for the application.
DMP_MemWrite_RCo(0116h, 0116h, LLh)
;         Check the correct Group Address Table Length
DMP_MemRead_RCo(0116h, 0116h)
;         If the received length is not equal to the programmed length,
;         abort the Configuration Procedure.
;         Clear Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, FFh)
;         Disconnect from the device
DMP_Disconnect_RCo
```

4.10.9.5.2 Local access

Not applicable.

4.10.9.5.3 Internal Access

Not applicable.

4.10.10 Group Address Table - E-Mode Realisation Type 2 (GrAT – Easy 2)

Used by: Ctrl-Mode reloc. DMA
and mask 0020h, 0021h and 0701h in E-Mode

4.10.10.1 Format

See clause 4.10.3.1.

4.10.10.2 Location

The location of the GrAT – Easy 2 is indicated by the pointer PID_TABLE_REFERENCE (PID = 7) of the Group Address Table Object. The object_index of the Group Address Table Object shall be 1 (please refer [09] “Appendix 5 : Controller Mode download Procedures”).

The Property PID_TABLE_REFERENCE shall be a read only Property.

4.10.10.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and Masks 0020h, 0021h and 0701h the following applies.

> A device receiving Link Services when Group Address Table, Group Object Association Table, Group Object Table and Application load states are different from *loaded* shall reject these services sending a negative response (start_index = 0 and no group_address_list), not depending on the communication mode.
>
> An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.10.10.4 Usage by the Management Server (device)

Please refer to clause 4.10.3.3 “Usage by the Management Server (device)”.

24) This makes the Data Link Layer does no longer pass L_Data-frames containing group communication to the upper layers.

4.10.10.5 Usage by the Management Client

4.10.10.5.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to read and to write the GrAT – Easy 2 and to get the maximum length of the GrAT – Easy 2.

4.10.10.5.1.1 Read GrAT – Easy 2

```
;       Connect to the device
DMP_Connect_RCo
;       Get GrAT – Easy 2 Table pointer
DMP_InterfaceObjectRead_R(object_index = 01h, PID = PID_TABLE_REFERENCE, start_index = 1,
        nr_of_elem = 1, GrAddrTabPtr)
;       Get Group Address Table Length
DMP_MemRead_RCo(GrAddrTabPtr, GrAddrTabPtr, AddressTableLength)
;       Read the Group Address entries of the Group Address Table.
;       Note that this management procedure foresees the optional splitting up
:       of the data in blocks of 12 octets.
DMP_MemRead_RCo(GrAddrTabPtr +3, GrAddrTabPtr +2*AddressTableLength, GrAT)
;       Disconnect from the device
DMP_Disonnect_RCo
```

4.10.10.5.1.2 Get maximum length GrAT – Easy 2

The calculation of the maximum length of the GrAT – Easy 2 is based on the Group Association Table following directly behind the GrAT – Easy 2 without space between the tables.

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Address Table pointer
DMP_InterfaceObjectRead_R(object_index = 01h, PID = PID_TABLE_REFERENCE, start_index = 1,
        nr_of_elem = 1, GrAddrTabPtr)
;       Get Group Association Table pointer
DMP_InterfaceObjectRead_R(object_index = 02h, PID = PID_TABLE_REFERENCE, start_index = 1,
        nr_of_elem = 1, GrAssocTabPtr)
;       Disconnect from the device
DMP_Disonnect_RCo
;       Calculate the maximum length Group Address Table Format 2:
;       ⇒       AddressTableMaxLength = (GrAssocTabPtr - GrAddrTabPtr - 4)/2 25)
```

25) The calculation is based on the memory block reserved for GrAT – Easy 2, including 4 octets for Table length, Individual Address and checksum. The Association Table is close to the Group Address Table (refer [09] "Appendix 5 : Controller Mode download Procedures").

4.10.10.5.1.3 Write GrAT – Easy 2

It shall be checked on beforehand by the Management Server if the download of the GrAT – Easy 2 does not conflict with other Resources [26]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Address Table pointer  27)
DMP_InterfaceObjectRead_R(object_index = 01h, PID = PID_TABLE_REFERENCE, start_index = 1,
        nr_of_elem = 1, GrAddrTabPtr)
;       Set Group Address Table "Loading" 28)
DMP_LoadStateMachineWrite_RCo_IO(object_index = 01h, PID = =PID_LOAD_STATE_CONTROL, start_index = 1,
        nr_of_elem = 1, Load)
;       Write the Group Address entries of the Group Address Table.
;       Note that this management procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemWrite_RCo(GrAddrTabPtr +3, GrAddrTabPtr +2*AddressTableLength, GrAT)
;       Set the Group Address Table length back to the value correct for the application.
DMP_MemWrite_RCo(GrAddrTabPtr, GrAddrTabPtr, LLh)
;       Check the correct Group Address Table Length
DMP_MemRead_RCo(GrAddrTabPtr, GrAddrTabPtr)
;       If the received length is not equal to the programmed length, abort the Configuration Procedure.
;       Set Group Address Table "Loaded"
DMP_LoadStateMachineWrite_RCo_IO(object_index = 01h, PID = PID_LOAD_STATE_CONTROL, start_index = 1,
        nr_of_elem = 1, LoadCompleted)
;       Disconnect from the device
DMP_Disonnect_Rco
```

4.10.10.5.2 Local Access

Not applicable.

4.10.10.5.3 Internal Access

Not applicable.

[26] Example: Group Address that have been inserted compared to the Group Address Table currently running in the Management Server, which would make the Group Objects Association Table invalid.

[27] Can be skipped if the pointer value is known from previous Management Procedures as for instance as specified in clause 4.10.10.5.1.2 “Get maximum length GrAT – Easy 2”.

[28] It is recommended to check the load via DMP_LoadStateMachineRead_RCo state before setting a new state. If the previous state is "Loading" it's the decision of the client to handle this state. All other states except "Loaded" shall stop the procedure.

4.11 Group Object Association Table (GrOAT)

4.11.1 Abstract Resource definition

The Group Objects Association Table shall be a parameter of the Application Layer. It shall store the relationship between Transport Layer Service Access Points (TSAPs) and Application Layer Service Access Points (ASAP). The information on this relationship is needed when mapping the Multicast Communication Mode messages A_GroupValue_Read, A_GroupValue_Response and A_GroupValue_Write to T_Data_Group-messages and vice versa. The TSAP shall be an index in the Group Address Table. The ASAP shall be the Group Object number. The lowest ASAP shall be 0.

The ASAP shall in this Resource be a unique identifier for a Group Object to the Application Layer. Please refer as well to [06]. The ASAP for this Resource shall thus be a Group Object Number.

Default state (optional)

The default state is not standardised but is implementation specific. This may be any of the following. Other default states are possible as well.

- The Group Object Association Table contains no associations; the Current Size is 0.
- The Group Object Association Table contains an implementation specific number of default associations; the size has an according value.

If the Group Object Association Table Realisation is controlled through a Load State Machine, then this may be Loaded or Unloaded.

4.11.2 Group Object Association Table - Realisation Type 1

Used by: Mask 0010h, 0011h, 0012h
Mask 0020h in compatibility mode , 0021h in compatibility mode
Mask 1012h, 1013h
Mask 3012h, 4012h configured in S-Mode

4.11.2.1 Format

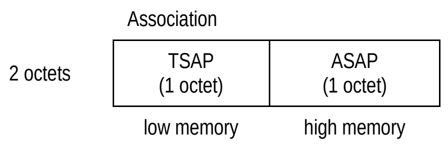
The Group Object Association Table shall be mapped in memory and shall have the following format.

	Low memory	Group Objects Association Table		Association Nr.
1 octet	TableAddress	Current Size		
2 octets	TableAddress +1	TSAP	ASAP	0
2 octets	TableAddress +3	TSAP	ASAP	1
2 octets	TableAddress +5	TSAP	ASAP	2
	…	…	…	…
…	High Memory			

GrAssocTabPtr = pointer to the Group Objects Association Table

The Current Size shall indicate the number Group Associations in the GrOAT.

An association shall always have the following format:



4.11.2.2 Location

The location of the Group Objects Association Table shall be given by the value contained in the Resource “Group Objects Association Table Pointer” (GrAssocTabPtr) incremented with an offset of 100h.

TableAddress = 100h + GrAssocTabPtr

GrAssocTabPtr

- Format

  Single octet value

- Location

  Address 111h

- Value

  Valid values are within the range 00h and FFh.

  Due to the location of other Resources (Group Address Table, EEPROM Checksum) this value can be only between 19h and FEh.

- Usage by Management Server

  The Management Server shall take the value given at this location incremented with 100h to find the location of the Group Objects Association Table.

- Usage by Management Client

  The Management Client shall set this value according to the location of the Group Objects Association Table.

4.11.2.3 Usage by Management Server (Device)

The algorithm for evaluation of the Group Objects Association Table shall depend on the communication way.

4.11.2.3.1 Frame transmission

When mapping a group message received from the Application Layer User to a T_Data_Group-message, the Application Layer shall map the contained ASAP to exactly one TSAP. To this, a “sending association” shall be defined.

The sending association shall be the Association with association number equal to the value of the ASAP. It is not checked whether the contained ASAP is the one for which the Application Layer User requested the transmission.

EXAMPLE ASAP nr. 4 is always transmitted using the association number 4.

4.11.2.3.2 Frame reception

The parameter TSAP, contained in the received T_Data_Group.ind and T_Data_Group.con services, shall be searched in the associations in the Group Objects Association Table. For every [29] instance of the TSAP that is found, the Application Layer

- shall take the ASAP that is associated to it by the association in which the TSAP is found, and
- shall generate the appropriate Application Layer service primitive [30], as specified in [06], using that ASAP as parameter, and
- shall stop the search for instances of the TSAP in the GrOAT when the end of the GrOAT is reached.

4.11.2.4 Usage by Management Client

4.11.2.4.1 Build rules

The evaluation mechanism by the Management Server leads to the following build rules for the Management Client.

- The size of the GrOAT shall at least be equal to the number of ASAPs.
- For each ASAP, the single TSAP on which its services shall be sent, shall be contained in the association with association number equal to the ASAP number.
- Non-sending associations can be put:
  - in the range of the sending associations (association numbers ≤ number ASAPs), if the TSAP is not used for a sending association with a higher association number.
  - in the range of the non-sending associations (association numbers > number of ASAPs)
- **Unused associations shall be indicated by a TSAP = FEh.**

4.11.2.4.2 Remote access

```
;	Connect to the device
DMP_Connect_RCo
;	Set Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, 00h)
;	Set Group Address Table Length to 1. 31)
DMP_MemWrite_RCo(0116h, 0116h, 01h)
;	Write the Group Object Association Table
;	Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemWrite_RCo(*GrAssocTabPtr, *GrAssocTabPtr+CurrentSize*2)
;	Set the GrAssocTabPtr to the correct value
DMP_MemWrite_RCo(0111h, 0111h, GrAssocTabPtr)
;	Set the Group Address Table length back to the value correct for the application.
DMP_MemWrite_RCo(0116h, 0116h, LLh)
;	Check the correct Group Address Table Length
DMP_MemRead_RCo(0116h,0116h)
;	If the received length is not equal to the programmed length, abort the Configuration Procedure.
;	Clear Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, FFh)
;	Restart the device
DMP_Restart
```

[29] This means that the reception of one T_Data_Group-service leads to as many A_Group_Value_Xxx-services as the TSAP is contained in the Group Object Association Table.

[30] A_GroupValue_Read.ind, A_GroupValue_Response.ind and A_GroupValue_Write.ind.

[31] This makes the Data Link Layer does no longer pass L_Data-frames containing group communication to the upper layers.

4.11.3 Group Object Association Table - Realisation Type 2

Used by: Mask 0020h, 0021h (both not in “compatibility mode”)

4.11.3.1 Format

The format is as specified in clause 4.11.2.1 “Format” above.

4.11.3.2 Location

The location of the Group Object Association Table – Realisation Type 2 shall be given by the value of the Property PID_TABLE_REFERENCE (PID = 7) of the System Interface Object “Group Object Association Table” (object_type = 02h). The Management Server (device) shall take care to set the PID_TABLE_REFERENCE to the value given by the Management Client when allocating the memory for this Resource.

4.11.3.3 Usage by the Management Server (device)

The algorithm for evaluation of the Group Objects Association Table shall depend on the communication way.

4.11.3.3.1 Frame transmission

When mapping a group message received from the Application Layer User to a T_Data_Group-message, the Application Layer shall map the contained ASAP to exactly one TSAP. To this, a "sending association" shall be defined.

The sending association shall be the Association with association number equal to the value of the ASAP. The Application Layer shall check whether the contained ASAP is the one for which the Application Layer User requested the transmission. If positive, the association shall be used for processing the transmission request. If negative, the transmission request shall be confirmed negatively by the Application Layer.

EXAMPLE ASAP nr. 4 is transmitted using the association number 4 if the ASAP entry in that association is equal to 4.

4.11.3.3.2 Frame reception

The usage by the Management Server for frame reception is as specified in clause 4.11.2.3.2 "Frame reception" above.

4.11.4 Group Object Association Table - Realisation Type 6

Used by: System 300
System B

4.11.4.1 Location

The Group Object Association Table – Realisation Type 6 shall be implemented as Interface Object of the type Associationtable Object (Object Type = 0002h).

For System B, this Interface Object shall provide the Property PID_TABLE_REFERENCE for memory mapped access to the table data. Please refer to [15] for the mandatory and optional Properties of this Interface Object in function of the Profile.

4.11.4.2 Format

The Group Object Association Table – Realisation Type 6 shall consist of one Interface Object of the type Associationtable Object (Object Type = 0002h). It shall provide the management operations for downloading. For the requirements about which Properties are mandatory and optional and for the access levels, please refer to Annex A in [15].

Table 45 - Association Table Interface Object

Property Name	Property Identifier	PDT (DPT)
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT
Interface Object Name	2 = PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[]
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL
Table Reference	07 = PID_TABLE_REFERENCE	PDT_UNSIGNED_LONG
Table	23 = PID_TABLE	PDT_GENERIC_02[] [a]
		PDT_GENERIC_04[] [a]
Memory Control Table	27 = PID_MCB_TABLE	PDT_GENERIC_08[]
Error code	28 = PID_ERROR_CODE	PDT_ENUM8
[a] It is mandatory to implement either one flavour of this Property.		

4.11.4.2.1 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This Property shall contain the Object Type of the Interface Object. The Associationtable Object shall have Object Type 0002h.

4.11.4.2.2 PID_OBJECT_NAME (PID = 2)

Please refer to 4.2.2 for the general requirements for PID_OBJECT_NAME.

This Property shall contain the name of the Association Table.

4.11.4.2.3 PID_LOAD_STATE_CONTROL (PID = 5)

Please refer to 4.2.5 for the general requirements for PID_LOAD_STATE_CONTROL.

4.11.4.2.4 PID_TABLE_REFERENCE (PID = 07)

Please refer to 4.2.7 for the general requirements for PID_TABLE_REFERENCE.

4.11.4.2.5 PID_TABLE (PID = 23)

This Property "Association Table" shall contain the actual Group Object Association Table which shall associate the Group Objects, through their Group Object Number (**A**pplication Layer **S**ervice **A**ccess **P**oint = ASAP) with the **T**ransport Layer **S**ervice **A**ccess **P**oint (TSAP). The TSAP shall be the index in the Group Address Table format 0 (S-Mode Group Address Table with Frame Format = 0) starting with index 1.

The Association Table can be of format 0 or format 1. The Management Client can determine which format is implemented via an A_PropertyDescription_Read. In the A_PropertyDescription_-Response-PDU the type field specifies either format 0 or format 1. This requires the implementation of full Interface Objects.

The format of the Group Object Association Table and the Property Datatype of PID_TABLE shall also be identical to the indication given by the field "Group Object Association Table Management" in the Property PID_MGT_DESCRIPTOR (see 4.3.23.3) if this Property is implemented.

The maximum number of elements in the Property shall show the maximum size of the table and current length shall show the current usage. In each association the high octet shall be the TSAP and the low octet shall be the ASAP number, both starting with value 1.

In case of transmission the first matching entry in the Association Table shall be used.

If the current length is set to 0 a “standard Association Table” shall be used (this is: ASAP = TSAP number). The Maximum Length shall be calculated via the load control relative allocation or shall be fixed in the device. The maximum length of the Property *Table* shall be read with an A_PropertyDescription_Read-service.

The content of the Property PID_TABLE shall only be valid in the load state ‘loaded’. Writing shall only have effect in the load state ‘Loading’.

PID_TABLE for the Group Object Association Table – format 0

In each association the high octet shall be the TSAP and the low octet shall be the ASAP, both starting with value 1.

Table 46 – PID_TABLE for the Group Object Association Table – format 0

Connection number (TSAP)	Group Object number (ASAP)	Property Array Index
Length (2 octet)		
TSAP #1 (1 octet)	ASAP #1 (1 octet)	1
TSAP #2 (1 octet)	ASAP #2 (1 octet)	2
TSAP #3 (1 octet)	ASAP #3 (1 octet)	3
…	…	…

PID_TABLE for the Group Object Association Table – format 1

In each association the most significant two octets shall be the TSAP and the least significant two octets shall be the ASAP number; both shall start with value 1.

Table 47 – PID_TABLE for the Group Object Association Table – format 1

Connection number (TSAP)	Group Object number (ASAP)	Property Array Index
Length (2 octets)		
TSAP #1 (2 octets)	ASAP #1 (2 octets)	1
TSAP #2 (2 octets)	ASAP #2 (2 octets)	2
TSAP #3 (2 octets)	ASAP #3 (2 octets)	3
…	…	…

Table 48 - Example for an Association Table

Property Array Index		**TSAP (starting with 1)**	**ASAP (starting with 1)**
0	Association table length (2 octets)	--	---
1	1. Group Association	1	1
2	2. Group Association	2	2
3	3. Group Association	1	5
….	….	….	

The sending ASAP shall be the first matching ASAP (like in System 7). No other sorting shall be required. If no association is found no request (in transmission) or update (in reception) shall be generated. For more details please refer to [06] clause 3.1 “The relation between TSAPs and ASAPs”.

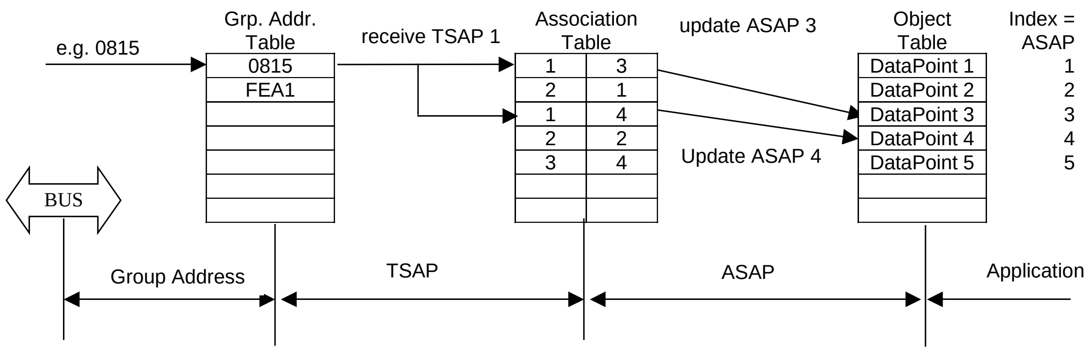


Figure 8 – Reception of a frame with Group Address 0815h (example)

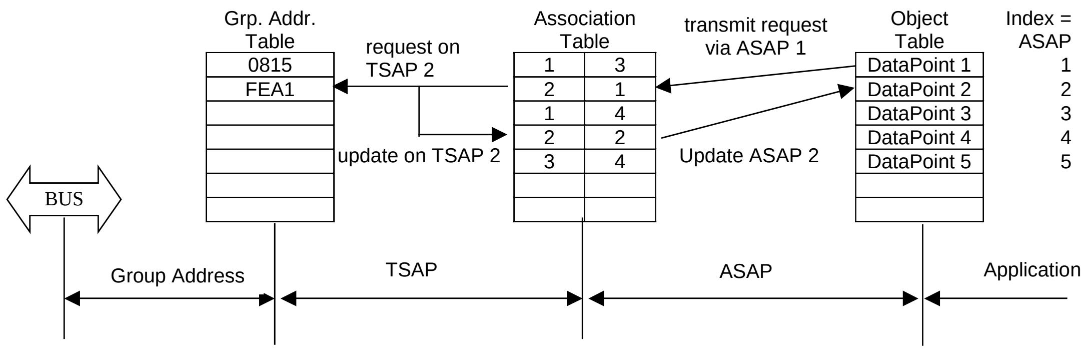


Figure 9 – Transmission of Group Object with ASAP 1

4.11.4.2.6 PID_MCB_TABLE (PID = 27)

Please refer to the Interface Object Type independent specification of PID_MCB_TABLE in 4.2.27.

For the Group Object Association Table object only the checksum feature is intended to be used but using the same mechanism as for the application program objects.

For this case the maximal table length (no. of elements) shall be set to ‘1’ and the read and write access shall be set to the lowest access level.

4.11.4.2.7 PID_ERROR_CODE (PID = 28)

Please refer to 4.2.28 for the general requirements for PID_ERROR_CODE.

4.11.4.3 Usage by the Management Client

4.11.4.3.1 General

The sequence is identical to the sequence as specified in 4.10.6.4 “Usage by the Management Client”.

For the Property Based Management no memory allocation is needed. The data shall be written directly to the Property Association Table. The maximum size of the table can be read via a description read in the maximum number of elements in the Association Table Property. To prevent the connectionless access to the Association Table it is recommended to not use the lowest access level for this Interface Object.

4.11.4.3.2 Remote Access

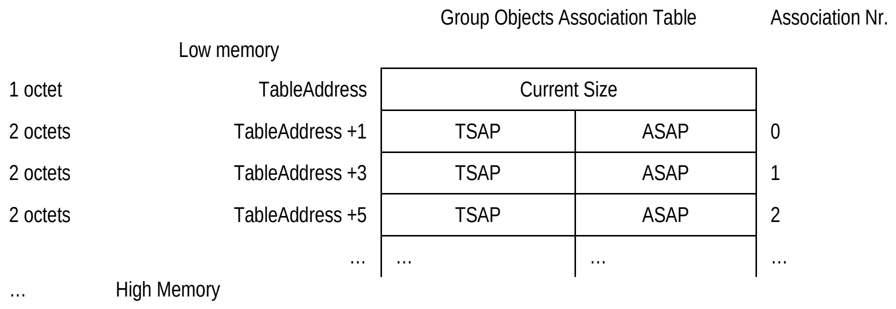
The following Load Controls shall be supported for the management of the Group Object Association Table - Realisation Type 6.

Load Control	Sub-type	Description	Supported
00h		No operation	M
01h		Start Loading	M
02h		Load Completed	M
03h		Additional Load Controls	
	0Ah	Relative Allocation	M
04h		Unload	M

4.11.5 Group Object Association Table - Realisation Type 8

Used by: Mask 5705h

4.11.5.1 Format



The Current Size shall indicate the number Group Associations in the GrOAT.

4.11.5.2 Location

The location of the Group Object Association Table – Realisation Type 8 - shall be given by the value of the Property PID_TABLE_REFERENCE (PID = 7) of the System Interface Object “Group Object Association Table” (object_type = 02h).

4.11.6 Group Object Association Table - E-Mode Realisation Type 1 (GrOAT – Easy 1)

Used by: Ctrl-Mode fixed DMA
and mask 0012h in E-Mode

4.11.6.1 Format

Please refer to clause 4.11.2.1 “Format”.

4.11.6.2 Location

The location of the GrOAT – Easy 1 shall be given by the value contained in the Resource “Group Objects Association Table Pointer” (GrAssocTabPtr) incremented with an offset of 100h.

TableAddress = 100h + GrAssocTabPtr

GrAssocTabPtr

- Format

  Single octet value

- Location

  Address 111h

- Value

  Valid values shall be within the range 00h and FFh.

  Due to the location of other Resources (Group Address Table, EEPROM Checksum) this value can be only between 19h and FEh.

- Usage by Management Server

  The Management Server shall take the value given at this location incremented with 100h to find the location of the GrOAT – Easy 1.

- Usage by Management Client

  The Management Client shall **read only** the value **to find out** the location of the GrOAT – Easy 1.

4.11.6.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and mask 0012h the following applies.

Before memory mapped access to a device, the E-Mode Controller shall set the error flags to 00h to stop the application. A device shall reject Link Services sending a negative response (start_index = 0 and no group_address_list) if error flags are set to 00h, not depending on communication mode (error flags definition is given in [18]).

An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.11.6.4 Usage by Management Server (Device)

Please refer to 4.11.2.3 “Usage by Management Server (Device)”.

4.11.6.5 Usage by Management Client

The evaluation mechanism by the Management Server leads to the following build rules for the Management Client.

- The size of the GrOAT – Easy 1 shall at least be equal to the number of ASAPs.
- For each ASAP, the single TSAP on which its services shall be sent, shall be contained in the association with association number equal to the ASAP number.
- Non-sending associations can be put:
  - in the range of the sending associations (association numbers ≤ number ASAPs), if the TSAP is not used for a sending association with a higher association number.
  - in the range of the non-sending associations (association numbers > number of ASAPs)

Unused sending associations shall be indicated by a TSAP = FEh.

Unused receiving associations shall be indicated by a TSAP = FDh.

4.11.6.5.1 Remote access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to read and to write the GrOAT – Easy 1 and to get the maximum length of the GrOAT – Easy 1.

4.11.6.5.1.1 Read GrOAT - Easy Format 1

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Association Table Pointer
DMP_MemRead_RCo(0111h, 0111h, GrAssocTabPtr)
;       Get Group Object Association Table Length
DMP_MemRead_RCo(0100h +GrAssocTabPtr, 0100h +GrAssocTabPtr, GrOAT_Length)
;       Read the Group Object Association Table
;       Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemRead_RCo(0100h +GrAssocTabPtr+1, 0100h +GrAssocTabPtr +2*GrOAT_Length, GrOAT)
;       Disconnect from the device
DMP_Disonnect_RCo
```

4.11.6.5.1.2 Get maximum length GrOAT – Easy 1

The calculation of the maximum length of the GrOAT – Easy 1 shall be based on the Group Object Table following directly behind the Association Table without space between the tables.

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Association Table Pointer
DMP_MemRead_RCo(0111h, 0111h, GrAssocTabPtr)
;       Get Group Object Table Pointer
DMP_MemRead_RCo(0112h, 0112h, GrTabPtr)
;       Disconnect from the device
DMP_Disonnect_RCo
;       Calculate the maximum length of Group Association Table Format 1:
;       ⇒       AssocTableMaxLength = (GrTabPtr - GrAssocTabPtr  -1)/2 32)
```

[32] The calculation is based on the memory block reserved for Association Table, including 1 octet for the “Length”. The Group Object Table is close to the Association Table (refer to the KNX Handbook Supplement 1 “Easy Configuration” Appendix 5)

4.11.6.5.1.3 Write GrOAT – Easy 1

It shall be checked on beforehand by the Management Server if the download of the GrOAT – Easy 1 does not conflict with other Resources [33]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;	Connect to the device
DMP_Connect_RCo
;	Get Group Object Association Table Pointer 34)
DMP_MemRead_RCo(0111h, 0111h, GrAssocTabPtr)
;	Set Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, 00h)
;	Set Group Address Table Length to 1 35)
DMP_MemWrite_RCo(0116h, 0116h, 01h)
;	Write the Group Object Association Table
;	Note that this management procedure foresees the optional splitting up
;	of the data in blocks of 12 octets.
DMP_MemWrite_RCo(0100h +GrAssocTabPtr, 0100h +GrAssocTabPtr +2*GrOAT_Length, GrOAT) 36)
;	Set the Group Address Table length back to the value correct for the application.
DMP_MemWrite_RCo(0116h, 0116h, LLh)
;	Check the correct Group Address Table Length
DMP_MemRead_RCo(0116h, 0116h)
;	If the received length is not equal to the programmed length, abort the Configuration Procedure.
;	Clear Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, FFh)
;	Disconnect from the device
DMP_Disconnect_RCo
```

4.11.6.5.2 Local Access

Not applicable.

4.11.6.5.3 Internal Access

Not applicable.

4.11.7 Group Object Association Table – E-Mode Realisation Type 2 (GrOAT – Easy 2)

Used by:	Ctrl-Mode reloc. DMA and mask 0020h, 0021h in E-Mode

4.11.7.1 Format

Please refer to clause 4.11.3.1 “Format”.

4.11.7.2 Location

The location of the GrOAT – Easy 2 shall be indicated by the pointer PID_TABLE_REFERENCE (PID = 7) of the Association table Object. The object_index of the Association table Object shall be 2 ([09] clause “Appendix 5: Controller Mode download Procedures”).

[33] Example:Associations that have been inserted compared to the Group Association Table currently running in the Management Server, which would make the Group Address Table invalid.

[34] Can be skipped if the pointer value is known from previous actions e.g. as in clause 4.11.6.5.1.2 “Get maximum length GrOAT – Easy 1”.

[35] This makes the Data Link Layer does no longer pass L_Data-frames containing group communication to the upper layers.

[36] This includes the new GrOAT_Length value.

The Property PID_TABLE_REFERENCE shall be a read only Property.

4.11.7.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and Masks 0020h, 0021h the following applies.

> A device receiving Link Services when Group Address Table, Group Object Association Table, Group Object Table and Application load states are different from *loaded* shall reject these services sending a negative response (start_index = 0 and no group_address_list), not depending on the communication mode.
>
> An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.11.7.4 Usage by the management server (device)

Please refer to 4.11.3.3.

4.11.7.5 Usage by the Management Client

The evaluation mechanism by the Management Server leads to the following build rules for the Management Client.

- The size of the GrOAT – Easy 2 shall at least be equal to the number of ASAPs.
- For each ASAP, the single TSAP on which its services shall be sent, shall be contained in the association with association number equal to the ASAP number.
- Non-sending associations can be put:
    - in the range of the sending associations (association numbers ≤ number ASAPs), if the TSAP is not used for a sending association with a higher association number.
    - in the range of the non-sending associations (association numbers > number of ASAPs)

Unused sending associations shall be indicated by a TSAP = FEh.

Unused receiving associations shall be indicated by a TSAP = FDh.

4.11.7.5.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to read and to write the GrOAT – Easy 2 and to get the maximum length of the GrOAT – Easy 2.

4.11.7.5.1.1 Read GrOAT – Easy 2

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Association Table pointer
DMP_InterfaceObjectRead_R(object_index = 02h, PID = PID_TABLE_REFERENCE, start_index = 1,
        nr_of_elem = 1, GrAssocTabPtr)
;       Get Group Object Association Table Length
DMP_MemRead_RCo(GrAssocTabPtr, GrAssocTabPtr, GrOAT_Length)
;       Read the Group Object Association Table.
;       Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemRead_RCo(GrAssocTabPtr +1, GrAssocTabPtr +2*GrOAT_Length, GrOAT)
;       Disconnect from the device
DMP_Disonnect_RCo
```

4.11.7.5.1.2 Get maximum length GrOAT – Easy 2

The calculation of the maximum length of the GrOAT – Easy 2 shall be based on the Group Object Table following directly behind the Association Table without space between the tables. The Group Object Table shall be the first part of the application program, the Group Object Table pointer shall match with the Property PID_TABLE_REFERENCE.

```
;         Connect to the device
DMP_Connect_RCo
;         Get Group Object Association Table Pointer
DMP_InterfaceObjectRead_R(object_index = 02h, PID = PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrAssocTabPtr)
;         Get Group Object Table Pointer
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrTabPtr)
;         Disconnect from the device
DMP_Disonnect_RCo
;         Calculate the maximum length Group Association Table Format 2:
;         ⇒        AssocTableMaxLength = (GrTabPtr - GrAssocTabPtr - 2)/2 37)
```

4.11.7.5.1.3 Write GrOAT – Easy 2

It shall be checked on beforehand by the management server if the download of the GrOAT – Easy 2 does not conflict with other Resources [38]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;         Connect to the device
DMP_Connect_RCo
;         Get Group Object Association Table pointer 39)
DMP_InterfaceObjectRead_R(object_index = 02h, PID = =PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrAssocTabPtr)
;         Set Group Object Association Table "Loading" 40)
DMP_LoadStateMachineWrite_RCo_IO(object_index = 02h, PID = =PID_LOAD_STATE_CONTROL, start_index = 1,
          nr_of_elem = 1, Load)
;         Write the Group Object Association Table.
;         Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemWrite_RCo(GrAssocTabPtr, GrAddrTabPtr +2* GrOAT_Length, GrOAT) 41)
;         Set Group Object Association Table "Loaded"
DMP_LoadStateMachineWrite_RCo_IO(object_index = 02h, PID = PID_LOAD_STATE_CONTROL, start_index = 1,
          nr_of_elem = 1, LoadCompleted)
;         Disconnect from the device
DMP_Disonnect_Rco
```

[37] The calculation is based on the memory block reserved for GrOAT – Easy 2, including 2 octets for the "Length" and checksum. The Group Object Table is close to the GrOAT – Easy 2 (refer to [09] Appendix 5 "Controller Mode download Procedures").

[38] Example: Associations that have been inserted compared to the GrOAT – Easy 2 currently running in the Management Server, which would make the GrOAT – Easy 2 invalid.

[39] Can be skipped if the pointer value is known from previous actions as for instance specified in 4.11.7.5.1.2 "Get maximum length GrOAT – Easy 2".

[40] It is recommended to check the load state via DMP_LoadStateMachineRead_RCo before setting a new state. If the previous state is "Loading" it's the decision of the client to handle this state. All other states except "Loaded" shall stop the procedure

[41] This includes the new GrOAT_Length value

4.11.7.5.2 Local Access

Not applicable.

4.11.7.5.3 Internal Access

Not applicable.

4.11.8 Group Object Association Table – E-Mode Realisation Type 3 (GrOAT – Easy 3)

Used by: Easy Ctrl reloc. DMA
and mask 0701h in E-Mode

4.11.8.1 Format

The format shall be identical to the format of the "Group Object Association Table – Realisation Type 1" as specified in 4.11.2.1 "Format".

4.11.8.2 Location

The location of the GrOAT – Easy 3 shall be indicated by the pointer PID_TABLE_REFERENCE (PID = 7) of the Association Table Object. The object_index of the Association Table Object shall be 2 (refer to [09] Appendix 5 "Ctrl-Mode download procedures").

The property PID_TABLE_REFERENCE shall be a read only Property.

4.11.8.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and Mask 0701h the following applies.

> A device receiving Link Services when Group Address Table, Group Object Association Table, Group Object Table and Application load states are different from *loaded* shall reject these services sending a negative response (start_index = 0 and no group_address_list), not depending on the communication mode.
>
> An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.11.8.4 Usage by the management server (device)

The algorithm for evaluation of the Group Objects Association Table shall depends on the communication way.

4.11.8.4.1 Frame transmission

When mapping a group message received from the Application Layer User to a T_Data_Group-message, the Application Layer shall map the contained ASAP to exactly one TSAP. To this, a "sending association" shall be defined.

The sending association shall be the first matching entry in the Group Object Association Table. The Application Layer shall check whether the contained ASAP is the one for which the Application Layer User requested the transmission. If positive, the association shall be used for processing the transmission request. If negative, the transmission request shall be confirmed negatively by the Application Layer.

4.11.8.4.2 Frame reception

The parameter TSAP, contained in the received T_Data_Group.ind and T_Data_Group.con services, shall be searched in the associations in the Group Objects Association Table. For every instance of the TSAP that is found, the Application Layer shall

- take the ASAP that is associated to it by the association in which the TSAP is found, and
- generate the appropriate Application Layer service primitive, as specified in [06], using that ASAP as parameter.

The search for instances of the TSAP in the GrOAT shall stop when the end of the GrOAT is reached.

4.11.8.5 Usage by the Management Client

The evaluation mechanism by the Management Server leads to the following build rules for the Management Client.

- The sorting of the table is different from mask 0020h, 0021h, 0012h.
- All entries in the Group Object Association Table may be used for sending, not only the first ones.
- If there are several connection numbers connected to a certain SAP the entry with the lowest association number shall be used. This means that the sending connection number shall be before all other connection numbers for a certain SAP.
- If there are several SAPs connected to one connection number the sending SAP shall send a message to the associated connection number and initiate an internal update to all other SAPs. In this case the sorting is of no importance.

4.11.8.5.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to read and to write the GrOAT – Easy 3 and to get the maximum length of the GrOAT – Easy 3.

4.11.8.5.1.1 Read GrOAT – Easy 3

```
;	Connect to the device
DMP_Connect_RCo
;	Get Group Object Association Table pointer
DMP_InterfaceObjectRead_R(object_index = 02h, PID = PID_TABLE_REFERENCE, start_index = 1,
	nr_of_elem = 1, GrAssocTabPtr)
;	Get Group Object Association Table Length
DMP_MemRead_RCo(GrAssocTabPtr, GrAssocTabPtr, GrOAT_CurrentSize)
;	Read the Group Object Association Table.
;	Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemRead_RCo(GrAssocTabPtr +1, GrAssocTabPtr +2*GrOAT_CurrentSize, GrOAT)
;	Disconnect from the device
DMP_Disonnect_RCo
```

4.11.8.5.1.2 Get maximum length GrOAT – Easy 3

The calculation of the maximum length of the GrOAT – Easy 3 shall be based on the Group Object Table following directly behind the Group Object Association Table without space between the tables. The Group Object Table shall be the first part of the Application Program, the Group Object Table pointer shall match with the property PID_TABLE_REFERENCE.

```
;         Connect to the device
DMP_Connect_RCo
;         Get Group Object Association Table Pointer
DMP_InterfaceObjectRead_R(object_index = 02h, PID = PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrAssocTabPtr)
;         Get Group Object Table Pointer
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrTabPtr)
;         Disconnect from the device
DMP_Disonnect_RCo
;         Calculate the maximum length Group Association Table Format 2:
;         ⇒         AssocTableMaxLength = (GrTabPtr - GrAssocTabPtr - 2)/2 42)
```

4.11.8.5.1.3 Write GrOAT – Easy 3

It shall be checked on beforehand by the Management Server if the download of the GrOAT – Easy 3 does not conflict with other Resources [43]. If this cannot be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;         Connect to the device
DMP_Connect_RCo
;         Get Group Object Association Table pointer 44)
DMP_InterfaceObjectRead_R(object_index = 02h, PID = =PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrAssocTabPtr)
;         Set Group Object Association Table "Loading" 45)
DMP_LoadStateMachineWrite_RCo_IO(object_index = 02h, PID = =PID_LOAD_STATE_CONTROL, start_index = 1,
          nr_of_elem = 1, Load)
;         Write the Group Object Association Table.
;         Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemWrite_RCo(GrAssocTabPtr, GrAddrTabPtr +2* GrOAT_CurrentSize, GrOAT) 46)
;         Set Group Object Association Table "Loaded".
DMP_LoadStateMachineWrite_RCo_IO(object_index = 02h, PID = PID_LOAD_STATE_CONTROL, start_index = 1,
          nr_of_elem = 1, LoadCompleted)
;         Disconnect from the device
DMP_Disonnect_Rco
```

[42] The calculation is based on the memory block reserved for GrOAT – Easy 3, including 2 octets for the current size and checksum. The Group Object Table is close to the GrOAT – Easy 3 (see [09] Appendix 5 “Ctrl-Mode download procedures”).

[43] Example: Associations that have been inserted compared to the GrOAT – Easy 3 currently running in the Management Server, which would make the GrOAT – Easy 3 invalid.

[44] Can be skipped if the pointer value is known from previous actions as for instance specified in 4.11.8.5.1.2

[45] It is recommended to check the load state via DMP_LoadStateMachineRead_RCo before setting a new state. If the previous state is "Loading" it's the decision of the client to handle this state. All other states except "Loaded" shall stop the procedure

[46] This includes the new GrOAT_CurrentSize value.

4.11.8.5.2 Local Access

Not applicable.

4.11.8.5.3 Internal Access

Not applicable.

4.12 Group Object Table

4.12.1 Abstract Resource Definition

The Group Object-Table shall contain general information about the internal Group Objects. It shall provide management operations for the internal Group Object Table. Elements of the Group Object Table shall be referred by the Association Table.

4.12.2 Group Object Table - Realisation Type 1

Used by:	Mask 0010h, 0011h, 0012h
	Mask 0020h in compatibility mode , 0021h in compatibility mode
	Mask 1012h, 1013h
	Mask 3012h, 4012h configured in S-Mode

4.12.2.1 Format

The Group Object Table is mapped in memory and has the following format:

	Low memory	Group Objects Table	Descriptor Nr.
1 octet	TableAddress	Current Size	
1 octet	TableAddress+1	RAM-Flags-Table Pointer	
3 octets	TableAddress+2	Group Object Descriptor	0
3 octets	TableAddress+5	Group Object Descriptor	1
3 octets	TableAddress+8	Group Object Descriptor	2
	…		…
…	High Memory		

The Current Size shall indicate the number of Group Objects in the GrOT.

4.12.2.1.1 RAM-Flags-Table Pointer

This single octet pointer contains the start address of the RAM-Flags-Table. This pointer shall point to User-RAM.

4.12.2.1.1.1 RAM-Flags-Table

The RAM-Flags Table forms the data interface between the user application and the Group Object Server.

- Format

low memory	RAM-Flags-Table	
Octet 0	Communication Flags Group Object 1 (4 bits)	Communication Flags Group Object 0 (4 bits)
Octet 1	Communication Flags Group Object 3 (4 bits)	Communication Flags Group Object 2 (4 bits)
…	…	…
high memory		

The communication flags are encoded as follows:

Bit	Name	Coding
3	update-flag	0 = not updated 1 = the value of the group object has been from the bus by an A_GroupValue_Write.ind or an A_GroupValue_Read.res.
2	data request flag	0 = idle/response 1 = data-request
1	transmission status	00 = idle/OK 01 = idle/error 10 = transmitting 11 = transmit request
0		

4.12.2.1.2 Group Object Descriptor

4.12.2.1.2.1 Format

A group object descriptor always has the following format:

Group Object Descriptor

3 octets	Data Pointer (1 octet)	Config Octet (1 octet)	Type Octet (1 octet)
	low memory		high memory

Data Pointer

The Data Pointer points to the MSB of the Group Object Value, that shall always start at octet boundary. Unused leading bits shall be set to 0. The address segment into which this pointer points is indicated by the “Segment Selector” field in the Config Octet, as specified below.

Config Octet

- Format: 1 octet

Bit	Field name
7	Shall be 1
6	Transmit Enable
5	Segment Selector Type
4	Write Enable
3	Read Enable
2	Communication Enable
1	Transmission Priority
0	

- Value

transmission priority: specifies the transmission-priority that is used for transmitting frames from this group object.

00	system priority
10	urgent priority
01	normal priority
11	low priority

communication enable: enables or disables communication via this group object

0	disabled	messages for this group object are not handled
1	enabled	messages for this group object are handled, in function of the other flags

read enable: specifies the behaviour of the Group Object server for this group object when an A_GroupValue_Read.ind is received for this group object.

0	disabled	the A_GroupValue_Read.ind service is ignored. no response is transmitted.
1	enabled	the group object server answers to an A_GroupValue_Read.ind to this group object with an A_GroupValue_Read.res containing this group objects value, under the condition that also the communication enable flag is set. It is possible to read the value from another internal object.

write enable: specifies the behaviour of the Group Object server for this group object when an A_GroupValue_Write.ind is received for this group object.

0	disabled	the A_GroupValue_Write.ind service is ignored. no update of the group object value.
1	enabled	the group object server updates the group object value with the value contained in the A_GroupValue_Write.ind service, under the condition that also the communication enable flag is set.

segment selector [47]: specifies the start address of the segment into which the data pointer points. The value of the data pointer has to be incremented with the value given below, in function of the segment selector, to obtain the address of the group object value.
This segment selector allows locating the group object value in RAM or in EEPROM.

0	segment = 00h
1	segment = 100h

[47] This was previously named “memory type”.

transmission enable: specifies the behaviour of the Group Object server when the transmit request flag is set (by the user application).

0	disabled	the transmit-request flag is ignored. No A_GroupValue_Write.req or A_GroupValue_Read.req is generated.
1	enabled	if the transmit request flag is set, it is evaluated and the appropriate Application Layer service primitive is generated, under the condition that the communication flag is set.

NOTE 28 In mask version 0010h, this field is not supported. The value of this bit shall be always 1.

Type Octet

This field specifies the size of the group object value [48].

- Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved. Shall be 0.	Reserved. Shall be 0.	Value-type					

- Coding

Type (dec.)	Value Size
0	1 bit
1	2 bits
2	3 bits
3	4 bits
4	5 bits
5	6 bits
6	7 bits
7	1 octet
8	2 octets
9	3 octets
10	4 octets
11	6 octets
12	8 octets
13	10 octets
14	14 octet

4.12.2.2 Location

The location of the Group Object Table – Realisation Type 1 is given by the value contained in the resource “Group Objects Table Pointer” (GrObjTabPtr) incremented with an offset of 100h.

TableAddress = 100h + GrObjTabPtr

[48] The formerly specified “VARDATA” size is no longer allowed.

GrObjTabPtr

- Format

  Single octet value

- Location

  Address 112h

- Value

  Valid values are within the range 00h and FFh.

  Due to the location of other resources (Group Address Table, EEPROM Checksum) this value can be only between 19h and FEh.

- Usage by Management Server

  The Management Server takes the value given at this location incremented with 100h to locate the Group Objects Association Table.

- Usage by Management Client

  The Management Client shall set this value according to the location of the Group Objects Association Table.

4.12.3 Group Object Table - Realisation Type 2

Used by: Mask 0020h, 0021h

4.12.3.1 Format

The format is identical as the “Group Object Table - Realisation Type 1” as specified in clause 4.12.2 above, except for the Group Object Config Octet.

Config Octet

- Format: 1 octet

This Config Octet Type contains an “Update Enable” field. The other fields are identical as above.

Bit	Name
7	Update Enable
6	Transmit Enable
5	Segment Selector Type
4	Write Enable
3	Read Enable
2	Communication Enable
1	Transmission Priority
0	

update enable:		specifies if the Group Object Value is updated if an A_GroupValue_Read.res service primitive is received for that Group Object.
0	disabled	the value of the Group Object is not updated. The update-flag in the RAM-Flags-Table for that Group Object is not set.
1	enabled	the value of the Group Object is updated by the value contained in the A_GroupValue_Read.res service primitive. The update-flag in the RAM-Flags-Table for that Group Object is set.

4.12.4 Group Object Table - Realisation Type 6

Used by: System 300

4.12.4.1 Location

The Group Object Table – Realisation Type 6 shall be implemented as Interface Object of the object type Group Object Table Object (Object Type = 0009h).

4.12.4.2 Format

The Group Object Table – Realisation Type 6 shall consist of one Interface Object of the object type Group Object Table (Object Type = 0009h) and shall have the Properties as specified in Table 54. It shall provide the management operations for downloading. For the requirements about which Properties are mandatory and optional and for the access levels, please refer to Annex A in [15].

Table 49 - Group Object Table Interface Object for Group Object Table – Realisation Type 6

Property Name	Property Identifier	PDT/DPT	
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	Group Object Table
Interface Object Name	2 = PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[ ]	Name of the Table
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	See description below (4.12.4.2.3)
Object Table	51 = PID_GRPOBJTABLE	PDT_GENERIC_06[]	Group Object Table
Extended Group Object Reference	52 = PID_EXT_GRPOBJREFERENCE	PDT_GENERIC_08[]	Group Object Reference

4.12.4.2.1 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This Property shall contain the Object Type of the Interface Object. The Group Object Table shall have Object Type 0009h.

4.12.4.2.2 PID_OBJECT_NAME (PID = 2)

Please refer to 4.2.2 for the general requirements for PID_OBJECT_NAME.

4.12.4.2.3 PID_LOAD_STATE_CONTROL (PID = 5)

For further information please refer to clause 4.17 “Load State Machine”. This Property is mandatory if the Application Program is loadable. If this Property does not exist the Group Object Table shall not be loadable.

4.12.4.2.4 PID_GRPOBJTABLE (PID = 51)

The Group Object Table shall be divided into two Properties.

The Property Object Table (PID_GRPOBJTABLE) shall include the relevant parts for the Management Client that shall be writeable.

The Property Extended Group Object Reference (PID_EXT_GRPOBJREFERENCE) shall include the internal reference to the data that may be readable and optionally writeable or completely hidden.

Array elements within the Properties PID_GRPOBJTABLE and PID_EXT_GRPOBJREFERENCE with the same index shall refer to the same Group Object.

The Property Object Table shall include the tool relevant part of an Extended Group Object. In this table every Group Object shall refer to a Property of an Interface Object of the device. This Property shall replace or be additional to the standard Group Object Table Property (PID_TABLE).

Format

Configuration	Datatype (High part of DPT)	Datatype (low part of DPT)
2 octets	2 octet	2 octet

Configuration

Bitnr	Name	Values	
0, 1	transmission priority	00b = system 10b = urgent 01b = normal 11b = low	Sets the transmission priority for a sending frame
2	communication enable	0 = disabled 1 = enabled	Enables the communication to the bus
3	read enable	0 = disabled 1 = enabled	Enables a read from bus
4	Write enable	0 = disabled 1 = enabled	Enables a write from bus
5	Memory segment	0 / 1	device internal memory segment selector
6	transmit enable	0 = disabled 1 = enabled	Enables a read/write to bus
7	response update enable	0 = disabled 1 = enabled	Update is used as a write if enable. (Responses from other devices leading to updates of Datapoints in own device)
8 to 15	Reserved	0	

Datatype

The field Datatype shall contain the 4 octet ($U_{16}U_{16}$) Datapoint Type identifier (DPT_ID) according which the Datapoint value is encoded. The data length to be transmitted is implicit in the high part of the Datapoint Type.

The Data point Type is as described [12]. E.g.:

1.001 DPT_Switch {Off=0, On=1}
…
5.001 DPT_Scaling {0.. 100%}
…
9.001 DPT_Value_Temp {273…+670760}…

4.12.4.2.5 PID_EXT_GRPOBJREFERENCE (PID = 52)

The Property Extended Group Object Reference shall include an Extended Group Object Table. In this table every Group Object shall refer to a Property of an Interface Object of the device. This Property shall be additional to the Group Object Table Property (PID_GRPOBJTABLE). The Property array element from this Property shall refer to the Property array element from PID_GRPOBJTABLE with the same start index. This Property shall describe the relation between Interface Object properties and Group Objects.

Format

2 octet	1 octet	1 octet	2 octet		1 octet	1 octet
Object Type	Object Instance	Property Identifier	Reserved (4 bit)	Start Index (12 bit)	Bit Offset	Conversion

Object Type

The Object Type shall include the type of the Interface Object that shall be equal to the value of the Object Type Property.

Object Instance

The Object Instance shall include the local Interface Object Instance. The Object Instance shall for each Object Type start with 1 and shall ascend with the local Interface Object index in the device.

Property Identifier

The Property Identifier shall be the identifier of the Property in the Interface Object referred by the Object Index.

Reserved

These 4 bits are reserved (always 0).

Start Index

This shall be the start index of the Property array (in case the Property data is no array, the start index shall be 1). This field shall be the start index in a Property and shall be 12 bit long. Only one array element of the Property can be accessed. Therefore the number of elements is fixed = 1 and not contained in the table.

Bit Offset

The Bit Offset shall be used to address single bits or only a part of a Property in a device. Bit offset shall start from "left" / MSB.

EXAMPLE From a structured Property with the type $U_{16}B_8$ the attribute Bit 0 of the B_8 field is mapped to a Boolean Group Object $\Rightarrow$ Bit Offset = 23

Conversion

This value shall identify a conversion function. Value 0 shall specify that there is no conversion. Values from 1 to 200 shall be reserved for standard conversions. Values from 201 to 254 are reserved for device specific conversions. Value 255 shall be reserved for escape.

A list of standard conversions will be defined for new application specific DPT in a separate document.

4.12.4.4 Usage by the Management Client

4.12.4.4.1 General

The sequence is identical to the sequence as specified in 4.10.6.4 "Usage by the Management Client".

4.12.4.4.2 Remote Access

The following Load Controls shall be supported for the management of the Group Object Table - Realisation Type 6.

Load Control	Subtype	Description	Supported
00h		No operation	M
01h		Start Loading	M
02h		Load Completed	M
03h		Additional Load Controls	
	0Ah	Relative Allocation	M
04h		Unload	M

4.12.4.4.3 Local Access

Local Access by a Management Client to the Group Address Table – Realisation Type 6 is not specified.

4.12.5 Group Object Table - Realisation Type 7

Used by: System B

4.12.5.1 Location

The Group Object Table – Realisation Type 7 shall be implemented as Interface Object of the object type Group Object Table Object (Object Type = 0009h).

4.12.5.2 Format

The Group Object Table – Realisation Type 7 shall contain general information about the internal Group Objects. It shall provide management operations for the internal Group Object Table. Elements of the Group Object Table shall be referred by the Association Table. For the requirements about which Properties are mandatory and optional and for the access levels, please refer to Annex A in [15].

Table 50 - Group Object Interface Object for Group Object Table – Realisation Type 7

Property Name	Property Identifier	PDT (DPT)	Description
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	Group Object Table Object
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	for further Information see Load / State- machines
Table Reference	07 = PID_TABLE_REFERENCE	PDT_UNSIGNED_LONG	This shall be a reference to the memory mapped access to the data of this Resource.
Table	23 = PID_TABLE	PDT_GENERIC_02[]	Group Object table Format 1
Memory Control Table	27 = PID_MCB_TABLE	PDT_GENERIC_08[]	Subsegmentation of memory space and check sum
Error code	28 = PID_ERROR_CODE	PDT_ENUM8	Specifies the reason for load state “ERROR”

4.12.5.2.1 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This Property shall contain the Object Type of the Interface Object. The Group Object Table shall have Object Type 0009h).

4.12.5.2.2 PID_LOAD_STATE_CONTROL (PID = 5)

For further information please refer to clause 4.17 “Load State Machine”.

4.12.5.2.3 PID_TABLE_REFERENCE (PID = 07)

Please refer to 4.2.7 for the general requirements for PID_TABLE_REFERENCE.

A Management Client shall use the value of this Property as reference for memory mapped access to the Group Object Table – Realisation Type 7.

4.12.5.2.4 PID_TABLE (PID = 23) – for Group Object Table – Realisation Type 7

4.12.5.2.4.1.1 Format

The Group Object Table Property format 1 shall include an array of Group Object descriptions.

The current length shall be the number of entries in the Group Object Table. The Maximum Length shall be calculated via the Load Control *Data Relative Allocation* or shall be fixed in the device.

The content of this Property shall only be valid in the load state *Loaded*. Writing shall only have effect in the load state *Loading*.

Table 51 - Group Object Table format

Property array index, i.e. the relative memory address		**ASAP**
0	Length (2 octets)	
1	Group Object descriptor (2 octets)	1
2	Group Object descriptor (2 octets)	2
n	…	n

Table 52 - Group Object Descriptor

Bit #	15	14	13	12	11	10	9	8
Meaning	Response Update enable 0 = disable 1 = enable	Transmit enable 0 = disable 1 = enable	Value Read on initialisation 0 = disable 1 = enable	Write enable 0 = disable 1 = enable	Read enable 0 = disable 1 = enable	Communi-cation enable 0 = disable 1 = enable	Transmission priority 00b = system 10b = urgent 01b = normal 11b = low	
Bit #	7	6	5	4	3	2	1	0
Meaning	Value Field Types							

4.12.5.2.4.1.2 Transmission Priority, Communication Enable, Read Enable, Write Enable, Transmit Enable and Response Update Enable

These fields shall have identical functionality and coding as the fields with the same name as specified in clauses 4.12.2.1.2.1 and 4.12.3.1.

4.12.5.2.4.1.3 Value read on initialisation

This shall specify whether or not the RAM-flags of the Group Object shall be set after reset of the application and consequently an A_GroupValue_Read shall be issued to update this Group Object value

0	disabled	The Group Object value shall not be updated after reset.
1	enabled	The Group Object value shall be updated after reset.

NOTE 29 ‘Value Read on Initialisation’ is a new feature introduced with System B.

As this feature has effect on busload especially after a reset of a complete installation this feature should be used very carefully (e.g. it should not be set by default in an average application).

4.12.5.2.4.1.4 Value Field Types

Table 53 - Value Field Types

Code	Symbol	Value size
0	Unsigned Integer (1)	1 bit
1	Unsigned Integer (1)	2 bit
2	Unsigned Integer (1)	3 bit
3	Unsigned Integer (1)	4 bit
4	Unsigned Integer (1)	5 bit
5	Unsigned Integer (1)	6 bit
6	Unsigned Integer (1)	7 bit
7	Octet 1	1 octet
8	Octet 2	2 octets
9	Octet 3	3 octets
10	Octet 4	4 octets
11	Octet 6	6 octets
12	Octet 8	8 octets
13	Octet 10	10 octets
14	Octet 14	14 octets
15	Octet 5	5 octets
16	Octet 7	7 octets
17	Octet 9	9 octets
18	Octet 11	11 octets
19	Octet 12	12 octets
20	Octet 13	13 octets
21	Octet 15	15 octets
22	Octet 16	16 octets
23	Octet 17	17 octets
…	…	…
254	Octet 248	248 octets
255	Octet 252	252 octets [49)]

[49)] Due to the fact that the frame length of an extended frame is encoded with a maximum value of 254 (255 = ESC Code) and that these maximum 254 octets cover the data plus two octets for the APCI and Transport Control the maximum data length is 252 octets.

4.12.5.2.5 PID_MCB_TABLE (PID = 27)

Please refer to clause 4.2.27 for the Interface Object Type independent specification of PID_MCB_TABLE.

This optional Property shall divide the segments ‘application program 1’ and ‘application program 2’ into multiple subsegments with access rights definable for each subsegment and carrying the checksum for the subsegments.

For the Group Object Interface Object only the checksum feature shall be used but using the same mechanism as for the Application Program objects.

For this case the table length (nr. of elements) shall be set to ‘1’ and the read and write access shall be set to the lowest access level.

The CRC checksum calculation shall provide a higher consistency while using the differential download.

The Management Server (device) shall properly calculate the CRC on the transition from the Load State *Loading* to the Load State *Loaded* according to the segment’s memory content right from beginning. Its value shall be valid in the load state *Loaded* only.

The 16 bit CRC shall be calculated according to the CRC16-CCITT specification.

For the specification of the Memory Control Block Table, please refer to 4.2.27 (Table 11).

For the specification of the CRC Control Byte please refer to clause 4.2.27.1.1.

4.12.5.2.6 PID_ERROR_CODE (PID = 28)

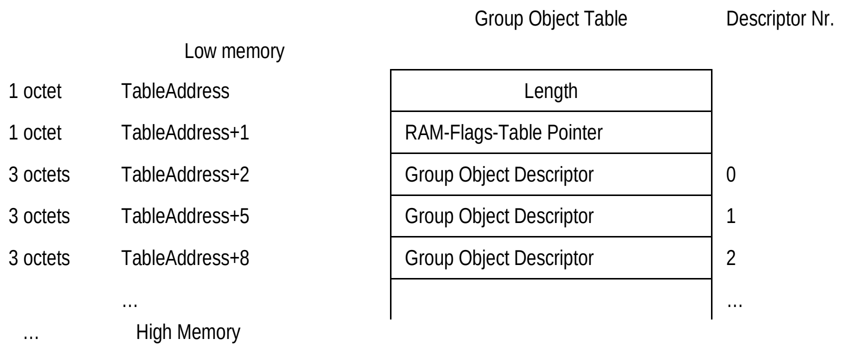
Please refer to 4.2.28 for the general requirements for PID_ERROR_CODE.

4.12.6 Group Object Table – E-Mode Realisation Type 1 (GrOT – Easy 1)

Used by: Ctrl-Mode fixed DMA
and mask 0012h in E-Mode

4.12.6.1 Format

The Group Object Table is mapped in memory and has the following format [50]:



The length shall indicate the number of Group Objects in the GrOT.

4.12.6.1.1 RAM-Flags-Table Pointer

Not applicable.

[50] The contents of the GrOT itself except the Length is not used by the Ctrl-Mode, but the general structure is needed for calculation of the absolute length of the GrOT.

4.12.6.1.2 Group Object Descriptor

Not applicable.

4.12.6.2 Location

The location of the GrOT – Easy 1 is given by the value contained in the Resource “Group Object Table Pointer” (GrTabPtr) incremented with an offset of 100h.

TableAddress = 100h + GrTabPtr

GrObjTabPtr

- Format

  Single octet value

- Location

  Address 112h

- Value

  Valid values are within the range 00h and FFh.

  Due to the location of other Resources (Group Address Table, EEPROM Checksum) this value can be only between 1Ah and FEh.

- Usage by Management Server

  The Management Server takes the value given at this location incremented with 100h to find the location of the GrOT – Easy 1.

- Usage by Management Client

  The Management Client shall **read only** the value **to find out** the location of the GrOT – Easy 1.

4.12.6.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and mask 0012h the following applies.

> Before memory mapped access to a device, the E-Mode Controller shall set the error flags to 00h to stop the application. A device shall reject Link Services sending a negative response (start_index = 0 and no group_address_list) if error flags are set to 00h, not depending on communication mode (error flags definition is given in [18]).
>
> An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.12.6.4 Usage by the Management Server (Device)

Please refer to 4.12.2.3 “Usage by the Management Server (Device)”.

4.12.6.5 Usage by the Management Client

4.12.6.5.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to get the pointer of the GrOT – Easy 1 [51].

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Table Pointer
DMP_MemRead_RCo(0112h, 0112h, GrTabPtr)
;       Disconnect from the device
DMP_Disonnect_RCo
;       Calculate the absolute pointer to Group Object Table Format 1:
;       ⇒       AbsGrTabPtr = 0100h +GrTabPtr
```

4.12.6.5.2 Local Access

Not applicable.

4.12.6.5.3 Internal Access

Not applicable.

4.12.7 Group Object Table – E-Mode Realisation Type 2 (GrOT – Easy 2)

Used by: Ctrl-Mode reloc. DMA
and mask 0020h, 0021h in E-Mode

4.12.7.1 Format

The format is identical to the "Group Object Table – E-Mode Realisation Type 1" as specified in clause 4.12.6.1 "Format" above.

4.12.7.2 Location

The location of the GrOT – Easy 2 shall be indicated by the pointer PID_TABLE_REFERENCE (PID = 7) of the Application Program Object. The object_index of the Application Program Object shall be 3. (Please refer to [09] Appendix 5 "Controller Mode download Procedures").

The property PID_TABLE_REFERENCE shall be a read only Property.

4.12.7.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and masks 0020h and 0021h the following applies.

> A device receiving Link Services when Group Address Table, Group Object Association Table, Group Object Table and Application load states are different from *loaded* shall reject these services sending a negative response (start_index = 0 and no group_address_list), not depending on the communication mode.

[51] The use of this procedure may be integrated into the calculation of the memory reference for easy parameter handling, as specified in clause 4.13.2.4.1 "Remote accessRemote access".

An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.12.7.4 Usage by the Management Server (Device)

Please refer to clause 4.12.3.3.

4.12.7.5 Usage by the Management Client

4.12.7.5.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any Management Client to get the pointer of the GrOT – Easy 2 [52].

```
;         Connect to the device
DMP_Connect_RCo
;         Get Group Object Table Pointer
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrTabPtr)
;         Disconnect from the device
DMP_Disonnect_RCo
```

4.12.7.5.2 Local Access

Not applicable.

4.12.7.5.3 Internal Access

Not applicable.

4.12.8 Group Object Table – E-Mode Realisation Type 3 (GrOT – Easy 3)

Used by: Easy Ctrl reloc. DMA
and mask 0701h in E-Mode

4.12.8.1 Format

The Group Object Table shall have the following format

	Address	Group Objects Table	Descriptor nr
1 octet	TableAddress	Current Size	
2 octets	TableAddress + 1	RAM-Flags-Table Pointer	
4 octets	TableAddress + 3	Group Object Descriptor	0
4 octets	TableAddress + 7	Group Object Descriptor	1
4 octets	TableAddress + 11	Group Object Descriptor	2
	…		…

The current size shall indicate the number of Group Objects in the GrOT.

[52] The use of this procedure may be integrated into the calculation of the memory reference for easy parameter handling, see 4.13.3.4.1.

4.12.8.2 Location

The location of the GrOT – Easy 3 shall be indicated by the pointer PID_TABLE_REFERENCE (PID = 7) of the Application Program Object. The object_index of the Application Program Object shall be 3. (please refer to [09] Appendix 5 “Ctrl-Mode download procedures”).

This Property shall point to the base address of the Group Object Table. This mandatory Property shall be set by the device . The Property PID_TABLE_REFERENCE shall be a read only Property. The address range shall be limited to 64k .

4.12.8.3 Resource data integrity in case of dual access by Link Services and memory mapped access

Please refer to the general requirements in 2.3.4.

In particular for the E-Mode Controller and masks 0701h the following applies.

> A device receiving Link Services when Group Address Table, Group Object Association Table, Group Object Table and Application load states are different from *loaded* shall reject these services sending a negative response (start_index = 0 and no group_address_list), not depending on the communication mode.
>
> An E-Mode Controller receiving rejected Link Services shall indicate in a proper way to the installer that the download procedure has failed.

4.12.8.4 Usage by the Management Server (Device)

This shall be identical as the usage by the Management Server for the Group Object Table - Realisation Type 2 as specified in 4.12.3.3.

4.12.8.5 Usage by the Management Client

4.12.8.5.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any management client to get the pointer of the GrOT – Easy 3 [53].

```
;         Connect to the device
DMP_Connect_RCo
;         Get Group Object Table Pointer
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_TABLE_REFERENCE, start_index = 1,
          nr_of_elem = 1, GrTabPtr)
;         Disconnect from the device
DMP_Disonnect_RCo
```

4.12.8.5.2 Local Access

Not applicable.

4.12.8.5.3 Internal Access

Not applicable.

[53] The use of this procedure may be integrated into the calculation of the memory reference for easy parameter handling, see 4.13.3.4.1.

4.13 Easy Parameter Block Table (PaBT)

4.13.1 Abstract Resource definition

Please refer to the [09] clause Appendix 6 “Structure for Parameters”/

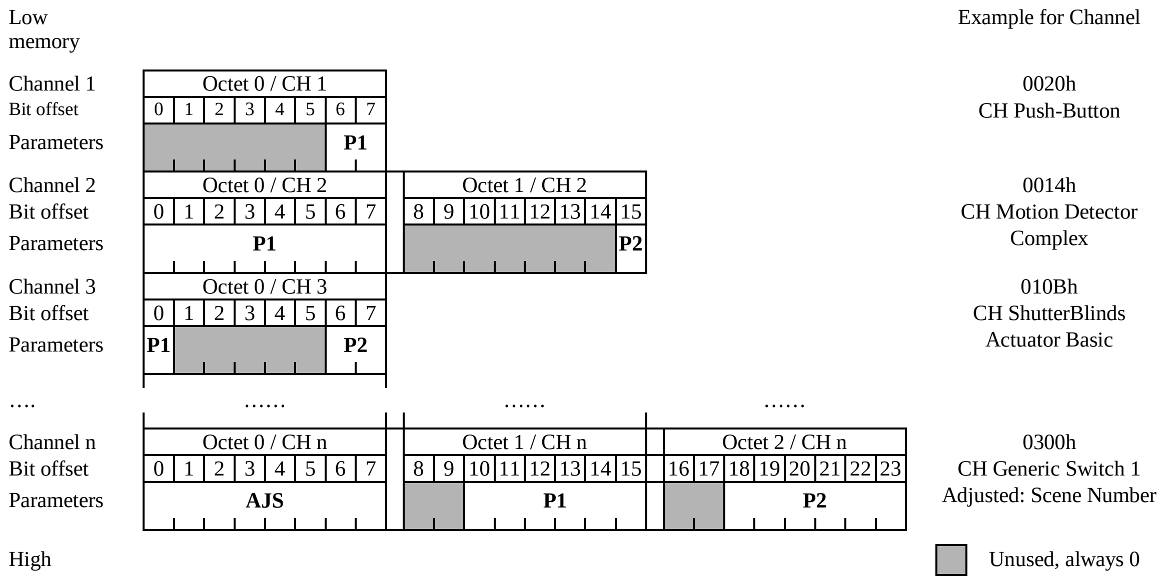
4.13.2 Parameter Block Table – E-Mode Realisation Type 1 (PaBT – Easy 1)

Used by: Ctrl-Mode fixed DMA
and mask 0012h in E-Mode

4.13.2.1 Format

Please refer to [09] Appendix 6 “Structures for Parameters”.

Format – Example (Informative)



4.13.2.2 Location

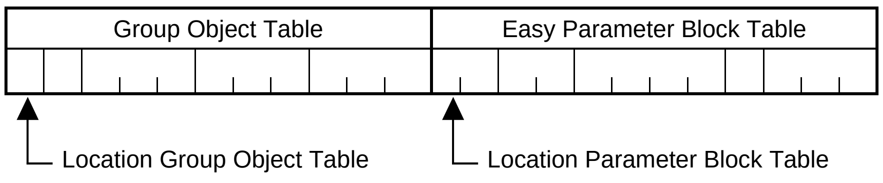
The location of the PaBT – Easy 1 is given by the value contained in the Resource “Group Object Table Pointer” (GrTabPtr) incremented by an offset of 100h and the Group Object Table "Length". The location is calculated based on these two Resources.

4.13.2.2.1 GrObjTabPtr

The pointer is as specified in clause 4.12.6.2 “Location above.

4.13.2.2.2 Parameter Block location

The PaBT – Easy 1 is located directly behind the Group Object Table, as specified in [09] Appendix 5 “Controller Mode download procedures”.


Group Object Table
Easy Parameter Block Table
Location Group Object Table
Location Parameter Block Table

4.13.2.2.3 Parameter Block calculation

ParaTabPtr = 100h + GrObjTabPtr + 3*GrOT_Length +2

4.13.2.3 Usage by the Management Server (device)

The Easy Parameter Block Table is used by the application running in the device in accordance to the supported E-Mode channels and the E-Mode Channel Code specification (please refer to [16] in the various Chapters with E-Mode Channel definitions). The table entries are normally read by the Application but in some cases depending on the supported E-Mode Channel the table entries are also written by the application (e.g. scene controller).

4.13.2.4 Usage by the Management Client

4.13.2.4.1 Remote access

The following Configuration Procedure specifies the standard sequence of Management Procedures that shall be followed by any Management Client to read and to write the PaBT – Easy 1.

4.13.2.4.1.1 Read PaBT – Easy 1

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Table Pointer
DMP_MemRead_RCo(0112h, 0112h, GrTabPtr)
;       Calculate the absolute pointer to Group Object Table Format 1
;       ⇒       AbsGrTabPtr = 0100h +GrTabPtr
;       Get Group Object Table Length
DMP_MemRead_RCo(AbsGrTabPtr, AbsGrTabPtr, GrOT_Length)
;       Calculate the absolute pointer to Parameter Block Table Easy Format 1
;       ⇒       AbsParaTabPtr = AbsGrTabPtr + 3*GrOT_Length +2
;       Read the Parameter Block Table Easy Format 1 54)
;       Note that this management procedure foresees the optional splitting up of the data in blocks of 12 octets.
DMP_MemRead_RCo(AbsParaTabPtr, AbsParaTabPtr + AbsParaTabLength 55) -1, PaBT)
;       Disconnect from the device
DMP_Disonnect_RCo
```

[54] The Easy Parameter Block Table can be read entirely or a single parameter value can be read from the device. In both cases, interpretation of a parameter in the complete read sequence or fetching a single parameter the position of the single parameter inside a parameter block shall be calculated as bit offset according the easy Channel code definition ([16] in the various Chapters with E-Mode Channel definitions) and [09] Appendix 5 “Controller Mode download procedures”.

[55] The parameter AbsParaTabLength is the sum of the parameter blocks of all channels in the device, counted in octets.

4.13.2.4.1.2 Write a Parameter [56] to PaBT – Easy 1

It shall be checked on beforehand by the Management Server if the writing to the Parameter Table does not conflict with other Resources [57]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;       Connect to the device
DMP_Connect_RCo
;       Get Group Object Table Pointer 58)
DMP_MemRead_RCo(0112h, 0112h, GrTabPtr)
;       Calculate the absolute pointer to Group Object Table Format 1
;       ⇒       AbsGrTabPtr = 0100h +GrTabPtr
;       Get Group Object Table Length 58)
DMP_MemRead_RCo(AbsGrTabPtr, AbsGrTabPtr, GrOT_Length)
;       Calculate the absolute pointer to PaBT – Easy 1 58)
;       ⇒       AbsParaTabPtr = AbsGrTabPtr + 3*GrOT_Length +2
;       Calculate the absolute parameter block location of the parameter block, where the parameter is in 59)
;       ⇒       AbsParaBlkAdr = AbsParaTabPtr + sum (previous channels in the device)
;       Calculate the absolute parameter location using the bitoffset according
;       the Easy Channel code definition 60)
;       ⇒       AbsParaAdr = AbsParaBlkPtr + bitoffset/8
;       Read the parameter value from the PaBT – Easy 1 61)
;       Note that this management procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemRead_RCo(AbsParaAdr, AbsParaAdr + ParaLength -1, ParaValue)
;       Mask the new parameter value to ParaValue, not affecting unused bits 61).
;       Set Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, 00h)
;       Write the parameter value to the PaBT – Easy 1
;       Note that this management procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemWrite_RCo(AbsParaAdr, AbsParaAdr + ParaLength 62) -1, ParaValue)
;       Clear Runtime Error Flags
DMP_MemWrite_RCo(010Dh, 010Dh, FFh)
;       Disconnect from the device
DMP_Disconnect_RCo
```

4.13.2.4.2 Local Access

Not applicable.

4.13.2.4.3 Internal Access

Not applicable.

[56] Parameters are typically written as a single action, depending on the installation procedures. This shall not prevent from writing the whole PaBT – Easy 1 or the parameters of a complete E-Mode channel in one step.

[57] Example:setting of an adjustable parameter in most cases make the Group Address Table and the Group Association Table invalid.

[58] Can be skipped if the pointer value is known from previous actions e.g. as specified in clause 4.13.2.4.1.1 “Read PaBT – Easy 1”.

[59] Minimal Information for E-Mode Channels is the block length, see [16] in the various Chapters with E-Mode Channel definitions.

[60] See [16] in the various Chapters with E-Mode Channel definitions.

[61] Can be skipped, if the parameter value starts and ends on an octet border (one or more complete octets).

[62] The length of the single parameter depends on the parameter type, see [16] in the various Chapters with E-Mode Channel definitions.

4.13.3 Parameter Block Table – E-Mode Realisation Type 2 (PaBT – Easy 2)

Used by: Easy Ctrl reloc. DMA
and Mask 0020h, 0021h in E-Mode

4.13.3.1 Format

The format is as specified in clause 4.13.2.1 “Format” above.

4.13.3.2 Location

The location of the PaBT – Easy 2 is given by the value contained in the Resource “Group Object Table Pointer” (PID_TABLE_REFERENCE in Object 3) and the Group Object Table "Length". The location is calculated based on these two Resources

4.13.3.2.1 GrObjTabPtr

The pointer is as specified in clause 4.12.6.2 “Location” above.

4.13.3.2.2 Parameter Block location

The Parameter Block location is as specified in clause 4.13.2.2.2 “Parameter Block location” above.

4.13.3.2.3 Parameter Block calculation

ParaTabPtr = GrObjTabPtr + 3*GrOT_Length +2

4.13.3.3 Usage by the Management Server (Device)

The usage is as specified in clause 4.13.2.3 “Usage by the Management Server (device)” above.

4.13.3.4 Usage by the Management Client

4.13.3.4.1 Remote Access

The following Configuration Procedure specifies the standard sequence of Management Procedures that shall be followed by any Management Client to read and to write the PaBT – Easy 2.

4.13.3.4.1.1 Read PaBT – Easy 2

```
;	Connect to the device
DMP_Connect_RCo
;	Get Group Object Table pointer
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_TABLE_REFERENCE, start_index = 1,
	nr_of_elem = 1, GrTabPtr)
;	Get Group Object Table Length
DMP_MemRead_RCo(GrTabPtr, GrTabPtr, GrOT_Length)
;	Calculate the absolute pointer to Parameter Block Table Easy Format 2:
;	⇒	AbsParaTabPtr = GrTabPtr + 3*GrOT_Length +2
;	Read the Parameter Block Table Easy Format 1 63)
;	Note that this management procedure foresees the optional splitting up
;	of the data in blocks of 12 octets.
DMP_MemRead_RCo(AbsParaTabPtr, AbsParaTabPtr + AbsParaTabLength 64) -1, PaBT)
;	Disconnect from the device
DMP_Disonnect_RCo
```

4.13.3.4.1.2 Write a Parameter [65] to PaBT – Easy 2

It shall be checked on beforehand by the Management Server if the download of the Group Address Table does not conflict with other Resources [66]. If this cannot be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;	Connect to the device
DMP_Connect_RCo
;	Get Group Object Table pointer 67)
DMP_InterfaceObjectRead_R(object_index = 03h, PID = =PID_TABLE_REFERENCE, start_index = 1,
	nr_of_elem = 1, GrTabPtr)
;	Get Group Object Table Length 67)
DMP_MemRead_RCo(GrTabPtr, GrTabPtr, GrOT_Length)
;	Calculate the absolute pointer to Parameter Block Table Easy Format 2 67)
;	⇒	AbsParaTabPtr = GrTabPtr + 3*GrOT_Length +2
;	Calculate the absolute parameter block location of the parameter block,
;	in which the parameter resides 68)
;	⇒ AbsParaBlkAdr = AbsParaTabPtr + sum (previous channels in the device)
;	Calculate the absolute parameter location using the bitoffset
;	according the Channel Code definition 69)
;	⇒	AbsParaAdr = AbsParaBlkPtr + bitoffset/8
;	Read the parameter value from the PaBT – Easy 2 70)
```

[63] The Easy Parameter Block Table can be read entirely or a single parameter value can be read from the device. In both cases, interpretation of a parameter in the complete read sequence or fetching a single parameter the position of the single parameter inside a parameter block shall be calculated as bit offset according [09] Appendix 5 “Ctrl-Mode download procedures”.

[64] The parameter AbsParaTabLength is the sum of the parameter blocks of all channels in the device, counted in octets.

[65] Parameters are typically written as single action, depending on the installation procedures. This shall not prevent to write the whole PaBT or the parameters of a complete E-Mode Channel in one step.

[66] Example:setting of an adjustable parameter in most cases make the Group Address Table and the Group Association Table invalid.

[67] Can be skipped if the pointer value is known from previous actions e.g. as specified in clause 4.13.3.4.1.1 “Read PaBT – Easy 2”.

[68] Minimal Information for E-Mode channels is the block length; please refer to [16] in the various Chapters with E-Mode Channel definitions.

[69] Please refer to [16] in the various Chapters with E-Mode Channel definitions.

```
;       Note that this management procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemRead_RCo(AbsParaAdr, AbsParaAdr + ParaLength 71) -1, ParaValue)
;       Mask the new parameter value to ParaValue, not affecting unused bits 70)
;       Set Application Program load state machine to "Loading" 72).
DMP_LoadStateMachineWrite_RCo_IO(object_index = 03h, PID = =PID_LOAD_STATE_CONTROL, start_index = 1,
        nr_of_elem = 1, Load)
;       Write the parameter value to the PaBT – Easy 2
;       Note that this management procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemWrite_RCo(AbsParaAdr, AbsParaAdr + ParaLength -1, ParaValue)
;       Set Application Program load state machine to "Loaded".
DMP_LoadStateMachineWrite_RCo_IO(object_index = 03h, PID = PID_LOAD_STATE_CONTROL, start_index = 1,
        nr_of_elem = 1, LoadCompleted)
;       Disconnect from the device
DMP_Disonnect_Rco
```

4.13.3.4.2 Local Access

Not applicable.

4.13.3.4.3 Internal Access

Not applicable.

[70] Can be skipped, if the parameter value starts and ends on an octet border (one or more complete octets).

[71] The length of the single parameter depends on the parameter type; please refer to the KNX Handbook Supplement 12 “Channel Codes”

[72] It is recommended to check the load state via DMP_LoadStateMachineRead_RCo before setting a new state. If the previous state is "Loading" it's the decision of the client to handle this state. All other states except "Loaded" shall stop the procedure

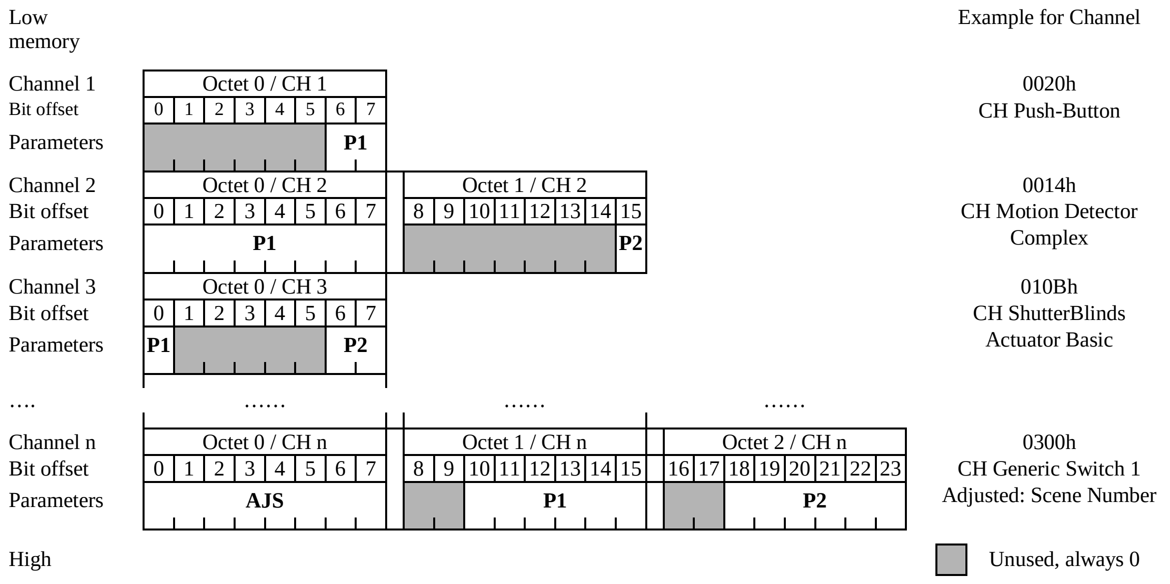
4.13.4 Parameter Block Table – E-Mode Realisation Type 3 (PaBT – Easy 3)

Used by: Easy Ctrl reloc. DMA
and mask 0701h in E-Mode

4.13.4.1 Format

Please refer to [09] “Appendix 6: Structures for Parameters”.

Format – Example (Informative)



4.13.4.2 Location

The location of the PaBT – Easy 3 shall be indicated by the pointer PID_PARAM_REFERENCE (PID = 51) of the Application Object. This Property shall point to base address of the Parameter Block. The object_index of the Application Object shall be 3.

The Property PID_PARAM_REFERENCE shall be a read only Property. This mandatory property shall be set by the device; it shall be of the type PDT_UNSIGNED_INT (16 bit) and shall be read only. The address range shall be limited to 64k.

4.13.4.3 Usage by the Management Server (Device)

The Easy Parameter Block Table shall be used by the application running in the device in accordance to the supported E-Mode Channels and the E-Mode Channel Code specification (please refer to [16] in the various Chapters specifying the E-Mode Channels). The table entries are normally read by the application but in some cases depending on the supported E-Mode Channel the table entries are also written by the application (e.g. scene controller).

4.13.4.4 Usage by the Management Client

4.13.4.4.1 Remote Access

The following Configuration Procedure specifies the standardized sequence of Management Procedures that shall be followed by any management client to read and to write the PaBT – Easy 3.

4.13.4.4.1.1 Read PaBT – Easy 3

```
;         Connect to the device
DMP_Connect_RCo
;         Get Parameter Block Pointer
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_PARAM_REFERENCE, start_index = 1,
          nr_of_elem = 1, PaBTPtr)
;         Read the Parameter Block Table E-Mode Format 1 [73]
;         Note that this Management Procedure foresees the optional splitting up
;         of the data in blocks of 12 octets.
DMP_MemRead_RCo(PaBTPtr, PaBTPtr + AbsParaTabLength [74] -1, PaBT)
;         Disconnect from the device
DMP_Disonnect_RCo
```

4.13.4.4.1.2 Write a Parameter [75] to PaBT – Easy 3

It shall be checked on beforehand by the Management Server if the download of the Group Address Table does not conflict with other Resources [76]. If this can not be excluded, the download of the other Resources shall be integrated in the following procedure.

```
;         Connect to the device
DMP_Connect_RCo
;         Get Parameter Block Pointer  [77]
DMP_InterfaceObjectRead_R(object_index = 03h, PID = PID_PARAM_REFERENCE, start_index = 1,
          nr_of_elem = 1, PaBTPtr)
;         Calculate the absolute parameter block location of the parameter block,
;         in which the parameter resides [78]
;         ⇒ AbsParaBlkAdr = PaBTPtr + sum (size of previous channel parameters in the device)
;         Calculate the absolute parameter location using the bit offset
;         according the E-Mode Channel Code definition [79]
;         ⇒ AbsParaAdr = AbsParaBlkPtr + bitoffset/8
```

[73] The E-Mode Parameter Block Table can be read entirely or a single parameter value can be read from the device. In both cases, interpretation of a parameter in the complete read sequence or fetching a single parameter the position of the single parameter inside a parameter block shall be calculated as bit offset according [09] clause “Appendix 5: Ctrl-Mode download procedures”.

[74] The parameter AbsParaTabLength shall be the sum of the parameter blocks of all E-Mode Channels in the device, counted in octets.

[75] Parameters are typically written as single action, depending on the installation procedures. This shall not prevent to write the whole PaBT or the parameters of a complete channel in one step.

[76] Example: setting of an adjustable parameter in most cases make the Group Address Table and the Group Association Table invalid.

[77] Can be skipped if the pointer value is known from previous actions e.g. as specified in clause 4.11.8.5.1.2.

[78] Minimal Information for E-Mode Channels is the block length; please refer [16] in the various Chapters specifying the E-Mode Channels.

[79] Please refer [16] in the various Chapters specifying the E-Mode Channels.

```
;       Read the parameter value from the PaBT – Easy 3 [80)]
;       Note that this Management Procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemRead_RCo(AbsParaAdr, AbsParaAdr + ParaLength [81)] -1, ParaValue)
;       Mask the new parameter value to ParaValue, not affecting unused bits.
;       Set Application Program load state machine to "Loading" [82)].
DMP_LoadStateMachineWrite_RCo_IO(object_index = 03h, PID = PID_LOAD_STATE_CONTROL, start_index = 1,
        nr_of_elem = 1, Load)
;       Write the parameter value to the PaBT – Easy 3
;       Note that this Management Procedure foresees the optional splitting up
;       of the data in blocks of 12 octets.
DMP_MemWrite_RCo(AbsParaAdr, AbsParaAdr + ParaLength -1, ParaValue)
;       Set Application Program load state machine to "Loaded".
DMP_LoadStateMachineWrite_RCo_IO(object_index = 03h, PID = PID_LOAD_STATE_CONTROL, start_index = 1,
        nr_of_elem = 1, LoadCompleted)
;       Disconnect from the device
DMP_Disonnect_Rco
```

4.13.4.4.2 Local Access

Not applicable.

4.13.4.4.3 Internal Access

Not applicable.

4.14 Application Program

4.14.1 Abstract Resource Definition

The Application Program Object shall contain global information about the internal user Application Program. It shall provide management operations for the internal user Application Program.

4.14.2 Application Program – Realisation Type 6

Used by: System 300

4.14.2.1 Location

The Application Program – Realisation Type 1 shall be implemented as Interface Object of the object type Application Program Object (Object Type = 0003h).

4.14.2.2 Format

The Application Program – Realisation Type 1 shall consist of one Interface Object of the object type Application Program Object (Object Type = 0003h) and shall have the Properties as specified in [15]. It shall provide the management operations for downloading.

All application specific parameters of the Application Program shall be set in application specific Interface Objects with the service A_PropertyValue_Write (See Application Interworking Standards in [16]).

80) Can be skipped, if the parameter value starts and ends on an octet border (one or more complete octets).

81) The length of the single parameter depends on the parameter type; please refer [16] in the various Chapters specifying the E-Mode Channels.

82) It is recommended to check the load state via DMP_LoadStateMachineRead_RCo before setting a new state. If the previous state is "Loading" it's the decision of the client to handle this state. All other states except "Loaded" shall stop the procedure.

Table 54 - Application program Interface Object

Property Name	Property Identifier	Type	Description
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	Application Object 3
Interface Object Name	2 = PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[]	Name of the Application program
Load Control	5 = PID_LOAD_STATE_-CONTROL	PDT_CONTROL	See description below
Run Control	6 = PID_RUN_STATE_-CONTROL	PDT_CONTROL	for further Information see Run / State machines
Application Version	13= PID_PROGRAM_-VERSION	PDT_GENERIC_05	Version of Application Program
PEI Type	16= PID_PEI_TYPE	PDT_UNSIGNED_CHAR	Required PEI-Type for User

4.14.2.2.1 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This Property shall contain the Object Type of the Interface Object. The Applicationprogram Object shall have Object Type 0003h.

4.14.2.2.2 PID_OBJECT_NAME (PID = 2)

Please refer to 4.2.2 for the general requirements for PID_OBJECT_NAME.

This Property shall contain the name of the Application Program.

4.14.2.2.3 PID_LOAD_STATE_CONTROL (PID = 5)

Please refer to 4.2.5 for the general requirements for PID_LOAD_STATE_CONTROL.

4.14.2.2.3.1 General

The sequence is identical to the sequence as specified in 4.10.6.4 “Usage by the Management Client”.

4.14.2.2.3.2 Remote Access

The following Load Controls shall be supported for the management of the Application Program – Realisation Type 6.

Load Control	Subtype	Description	Supported
00h		No operation	M
01h		Start Loading	M
02h		Load Completed	M
04h		Unload	M

4.14.3 Application Program – Realisation Type 7

Used by: System B

4.14.3.1 Use

The Application Program – Realisation Type 7 is named “Application Program 1” in System B. It shall contain global information about the internal user Application Program 1. It shall provide management operations for the internal user Application Program 1.

4.14.3.2 Location

The Application Program 1 Object shall be implemented as Interface Object of the object type *Applicationprogam Object* (Object Type = 0003h).

4.14.3.3 Format

4.14.3.3.1 Overview

Table 55 - Application Program 1 Interface Object

Property Name	Property Identifier	PDT/DPT	Description
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	Applicationprogram Object
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	See description below
Run Control	6 = PID_RUN_STATE_CONTROL	PDT_CONTROL	See 4.18.2 “Run State Machine – Realisation Type 1 (Property based)”
Application Version	13 = PID_PROGRAM_VERSION	PDT_GENERIC_05	Version of Application Program 1
PEI-Type	16 = PID_PEI_TYPE	PDT_UNSIGNED_CHAR	Required PEI-Type for User
Table Reference	07 = PID_TABLE_REFERENCE	PDT_UNSIGNED_LONG	Pointer to Application Program 1 Base Address
Memory Control Table	27 = PID_MCB_TABLE	PDT_GENERIC_08[]	Subsegmentation of memory space and checksum
Error code	28 = PID_ERROR_CODE	PDT_ENUM8	Specifies the reason for Load State “ERROR”

4.14.3.3.2 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This shall be the Object Type of the Interface Object (Application Program 1 Object = 0003h). This property is always mandatory and shall always be readable.

4.14.3.3.3 PID_LOAD_STATE_CONTROL (PID = 5)

Please refer to clause 4.2.5 for the specification of PID_LOAD_STATE_CONTROL and to clause 4.17 for the specification of the Load State Machine.

4.14.3.3.4 PID_RUN_STATE_CONTROL (PID = 6)

Please refer to clause 4.2.6 for the specification of PID_RUN_STATE_CONTROL and to clause 4.18.2 for the specification of the Run State Machine.

The Property Run Control shall control the Application Program 1 of the device.

The minimum implemented possible states and the run state machine define state changes of an executable part.

4.14.3.3.5 PID_PEI_TYPE (PID = 16)

Please refer to clause 4.2.16 for the specification of PID_PEI_TYPE.

4.14.3.3.6 PID_TABLE_REFERENCE (PID = 07)

Please refer to clause 4.2.7 for the specification of PID_TABLE_REFERENCE.

When the Application Program 1 object is *Unloaded* then the Management Server shall set the value to zero. When the Management Client has successfully allocated memory then the value shall be set to the absolute memory location from where the table will be downloaded. When memory allocation is not successful then the value shall be set to zero.

4.14.3.3.7 PID_MCB_TABLE (PID = 27)

Please refer to clause 4.2.27 for the Interface Object Type independent specification of PID_MCB_TABLE.

This optional Property shall divide the segment into multiple subsegments with access rights definable for each subsegment and carrying the checksum for the subsegments.

The use case for this subsegmentation is to separate code and parameter sections. A code section is normally protected with the highest access level; a parameter section with a lower access level. The maximum length of this table (nr. of elements) shall be fixed within the device and shall be read with an A_PropertyDescription_Read. The minimum length is ‘1’.

For the specification of the Memory Control Block Table, please refer to 4.2.27 (Table 11).

For the specification of the CRC Control Byte please refer to clause 4.2.27.1.1.

4.14.3.3.8 PID_ERROR_CODE (PID = 28)

Please refer to clause 4.2.28.

4.15 PEI Program

4.15.2 PEI Program – Realisation Type 7

Used by: System B

4.15.2.1 Use

The PEI Program – Realisation Type 7 is named *Application Program 2* in System B. It shall contain global information about the internal Application Program 2. It shall provide management operations for the internal Application Program 2.

4.15.2.2 Location

The Application Program 2 Object shall be implemented as Interface Object of the object type *Interfaceprogram Object* (Object Type = 00034).

4.15.2.3 Format

4.15.2.3.1 Overview

Table 56 - Application Program 2 Interface Object

Property Name	Property Identifier	PDT/DPT	Description
Interface Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	Interfaceprogram Object
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	See description below
Run Control	6 = PID_RUN_STATE_CONTROL	PDT_CONTROL	for further Information see Run / State machines
Application Version	13 = PID_PROGRAM_VERSION	PDT_GENERIC_05	Version of Application Program
PEI-Type Required	16 = PID_PEI_TYPE	PDT_UNSIGNED_CHAR	Required PEI-Type for User
Pointer to base address Format 1	07 = PID_TABLE_REFERENCE	PDT_UNSIGNED_LONG	Pointer to address of Application Program 2 base
Memory Control Table	27 = PID_MCB_TABLE	PDT_GENERIC_08[]	Sub-Segmentation of memory space and check sum
Error code	28 = PID_ERROR_CODE	PDT_ENUM8	Specifies the reason for load state “ERROR”

4.15.2.4 PID_OBJECT_TYPE (PID = 1)

Please refer to 4.2.1 for the general requirements for PID_OBJECT_TYPE.

This shall be the Object Type of the Interface Object (Application Program 2 Object = 0004h). This Property is always mandatory and shall always be readable.

4.15.2.5 PID_LOAD_STATE_CONTROL (PID = 5)

Please refer to clause 4.2.5 for the specification of PID_LOAD_STATE_CONTROL and to clause 4.17 for the specification of the Load State Machine.

4.15.2.6 PID_RUN_STATE_CONTROL (PID = 6)

Please refer to clause 4.2.6 for the specification of PID_RUN_STATE_CONTROL and to clause 4.18.2 for the specification of the Run State Machine.

The transitions between the ‘Ready’ and the ‘Running’ states shall always be made automatically by the system depending on the run conditions The states in the device that have to be reached to fulfil the run conditions depend upon the application and are, therefore, not standardized. For example it could be possible that the same or different run conditions apply for application program 1 and application program 2. It is manufacturer’s responsibility to take care that the device cannot reach a critical state when the application program is started.

For diagnostic purposes, it shall be possible to start and stop Application Program 2 (if the run conditions are fulfilled).

4.15.2.7 PID_TABLE_REFERENCE (PID = 07)

Please refer to clause 4.2.7 for the specification of PID_TABLE_REFERENCE.

When the Application Program 2 object is Unloaded then the Management Server shall set the value to zero. When the Management Client has successfully allocated memory then the value shall be set to the absolute memory location from where the table will be downloaded. When memory allocation is not successful then the value shall be set to zero.

4.15.2.8 PID_MCB_TABLE (PID = 27)

Please refer to clause 4.2.27 for the Interface Object Type independent specification of PID_MCB_TABLE.

This optional Property shall divide the segment into multiple subsegments with access rights definable for each subsegment and carrying the checksum for the subsegments.

The use case for this subsegmentation is to separate code and parameter sections. A code section is normally protected with the highest access-level a parameter section with a lower access-level. The maximal length of this table (nr. of elements) shall be fixed within the device and shall be read with an A_PropertyDescription_Read. The minimum length is ‘1’.

For the specification of the Memory Control Block Table, please refer to 4.2.27 (Table 11).

For the specification of the CRC Control Byte please refer to clause 4.2.27.1.1.

4.15.2.9 PID_ERROR_CODE (PID = 28)

Please refer to clause 4.2.28.

4.16 KNX Serial Number

4.16.1 Abstract Resource Definition

4.16.1.1 Abstract Resource definition

The KNX Serial Number shall be a unique identifier for the device.

Whether or not a device shall support a KNX Serial Number can be derived from [15].

The owner of the micro controller shall ensure the global uniqueness of the leading four octets within the specific manufacturer’s code space.

4.16.1.2 Format

Regardless to the Realisation Type, the KNX Serial Number shall be encoded according the specification of DPT_SerNum (DPT_ID = 221.001; see [12]). The Property Datapoint Type shall be PDT_GENERIC_06.

This is illustrated by the frame examples underneath [83].

6		Octet 7								Octet 8								Octet 9								Octet 10								Octet 11								Octet 12								Octet 13							
°)										Manufacturer code								Manufacturer code								Number incremented with each BAU																															
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	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
APCI	APCI	APCI	APCI	APCI	APCI	APCI	APCI	APCI	APCI																																																
1	1	1	1	0	1	1	1	0	0	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

°) last two bits of octet 6

Figure 10 - A_IndividualAddress_SerialNumber_Read-PDU (example)

[83] The indicated octet numbers are valid for TP1.

Octet 6								Octet 7								Octet 8								Octet 9								Octet 10								Octet 11							
						APCI										Object_index								Property_id								nr_of_elem				start_index											
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
						APCI	APCI	APCI	APCI	APCI	APCI	APCI	APCI	APCI	APCI																																
						1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Octet 12 - Manufacturer Code								Octet 13 - Manufacturer Code								Octet 14 – 17 : Number incremented with each BAU																															
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
m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Figure 11 - A_PropertyDescription_Response-PDU (example)

4.16.1.3 Usage by the Management Server (device) – general

The support of the KNX Serial Number may encompass the support of the following features:

- KNX Serial Number Realisation Type 1 (memory mapped; see 4.16.2), and/or
- KNX Serial Number Realisation Type 2 (Property; see 4.16.3), and/or
- the AL-services A_IndividualAddress-SerialNumber_Read and A_IndividualAddressSerialNumber_Write

However, these features cannot be combined in any combination. Instead, the following rules apply.

1. If Interface Objects (reduced or full) are supported and the AL services A_IndividualAddress-SerialNumber_Read and A_IndividualAddressSerialNumber_Write are supported too, then the Property PID_SERIAL_NUMBER shall be present in the device.
2. If Interface Objects (reduced or full) are supported and the Property PID_SERIAL_NUMBER is present then the AL services A_IndividualAddress-SerialNumber_Read and A_IndividualAddressSerialNumber_Write shall be supported too.
3. Devices with memory mapped KNX Serial Number may support the AL services A_Individual-AddressSerialNumber_Read and A_IndividualAddressSerialNumber_Write without the support of the Property PID_SERIAL_NUMBER. These devices have no Interface Object server.

The allowed combinations are specified for each relevant Profile in [15]. If no explicit definition is available then these default rules shall apply.

4.16.2 KNX Serial Number - Realisation Type 1

4.16.2.1 Format

Please refer to 4.16.1.2.

4.16.2.2 Location

In this KNX Serial Number – Realisation Type 1, the KNX Serial Number is exclusively accessible via accessible via the Application Layer services A_IndividualAddressSerialNumber_Read and A_IndividualAddressSerialNumber_Write. It is not accessible via memory mapped access, via a Property in an Interface Object or any other.

4.16.3 KNX Serial Number - Realisation Type 2

4.16.3.1 Format

Please refer to 4.16.1.2.

4.16.3.2 Location

For the Realisation Type 2, the KNX Serial Number is realised as PID_SERIAL_NUMBER; please refer to 4.2.11.

4.17 Load State Machine

4.17.1 Abstract Resource definition

The Load State Machine shall be able to manage all kinds of downloadable configuration data including executable code. More than one Load State Machine is possible in one device.

A Load State Machine for executable code is possibly combined with a Run State Machine. The state of the Load State Machine shall be stored in non-volatile memory (e.g. EEPROM, flash).

4.17.2 Load State Machine – Realisation Type 1 (Property based)

4.17.2.1 Format

Realisation Type 1 for the Load State Machine shall be a Property of Property Datatype PDT_CONTROL. The format shall differ between the read access, as specified in 2012.10.22: VL ←AN149 “New Load State LoadCompleting” clause 2.3.1 [20121022b].

Table 57 and write access, as specified in Table 58.

The transitions between the states shall be less than 30 seconds. This time shall be the minimum delay during which a MaC shall wait after sending a load event until interpretation of the load event with the expected state transition to be failed.

4.17.2.2 Location

The load procedure for device configuration data specifies a general mechanism for changing the different programmable parts of a device. Each programmable part shall have exactly one dedicated Load State Machine that controls the load process. This Load State Machine shall be accessed by an associated Interface Object with the Property Load Control (PID_LOAD_STATE_CONTROL, see also 4.2.5).

The Load Procedure for device configuration data specifies a general mechanism for changing the different loadable parts of a device. Each loadable part shall be represented by exactly one dedicated Interface Object. The Load State Machine of the loadable part shall be accessed exclusively through the dedicated Property PID_LOAD_STATE_CONTROL within that Interface Object.

The value of the Property Load Control shall be stored in non-volatile memory, because it shall be preserved also on power fail.

4.17.2.3 Usage by the Management Server (device)

4.17.2.3.1 States of the Load State Machine

Each Property Load Control shall provide the state of the Load State Machine by providing read access to the Property value. The load states (data read from the Property Load Control) shall be encoded as follows.

Table 57 - Load States

Load State	Value	M/O [a]	Remark
Unloaded	0	M	No data is loaded. The data that is associated with the Property Load Control is not valid (e.g.: Address Table is not valid).
Loaded	1	M	Valid data is loaded. Only in this state the associated data shall be considered as valid; in all other states the data shall be considered as invalid
Loading	2	M	Load process is active.
Error	3	M	Error in data detected or error during load process.
Unloading	4	O	Unload process is active
LoadCompleting	5	O	Intermediate state between Loading and Loaded, e.g. checksum calculation process is active
[a] M: mandatory O: optional			

4.17.2.3.2 Events of the Load State Machine

There are two types of events that can cause a transition in the Load State Machine:

1. external events are write accesses to the Property Load Control by a Management Client, and
2. internal events in the Management Server.

The reaction of the Management Server is already partly specified under 4.17.2.1 “Format” above.

Illegal additional events (e.g. invalid values written to the Property) shall be ignored and shall not lead to a change of state.

The Load State Machine shall enter the state Error in case of an error. Unknown events shall be ignored.

The event *UnloadCompleted* is mostly only an internal event. Therefore, a Management Client may never observe the state Unloading.

The event *LoadCompletingCompleted* is only an internal event. The event LoadCompleted may cause the Load State Machine to change firstly to the intermediate state LoadCompleting and then, by the internal event LoadCompletingCompleted to state Loaded. For a Load State Machine supporting state LoadCompleting, a MaC may observe the state LoadCompleting.

If the Management Server (MaS) goes offline during the state LoadCompleting, the Load State Machine shall send state LoadCompleting at least once before going offline, i.e. with the A_PropertyValue_Response PDU sent as reaction to the received A_PropertyValue_Write PDU containing the load event LoadCompleted.

Write accesses to the Property Load Control shall be events to the Load State Machine. The encoding of the different Load Events (data written to the Property Load Control) shall be as specified in Table 58.

Table 58 - Load Events

Load Event	Value	Reaction of the Load State Machine
No Operation	00h	Nothing
Start Loading	01h	This shall be interpreted as the request to start the loading of the loadable part.
Load Completed	02h	This shall be interpreted as the request to complete the Loading of the loadable part if this event occurs during the state Loading. Possible checksum shall be calculated, all data shall be declared as valid and the Load State shall change to the state Loaded, if no internal errors occurs and no error in the data is detected. If the transition from Loading to Loaded takes more than 2 seconds, e.g. by complex checksum calculation, the Load State shall first change to intermediate state LoadCompleting and change to Loaded after calculation is finished.
Additional Load Controls	03h	This Load Event shall be used to transfer additional load information like memory allocation (absolute or relative), application entry points or special run conditions for an application. See clause 3.27 “DM_LoadStateMachineWrite” in [08].
Unload	04h	This Load Event shall be interpreted as the request to unload the loadable part. The loadable data shall be declared as invalid. The data is undefined.

4.17.2.3.3 Load State Machine

The state transition diagram as specified in Figure 12 shall be valid for all Load State Machines.

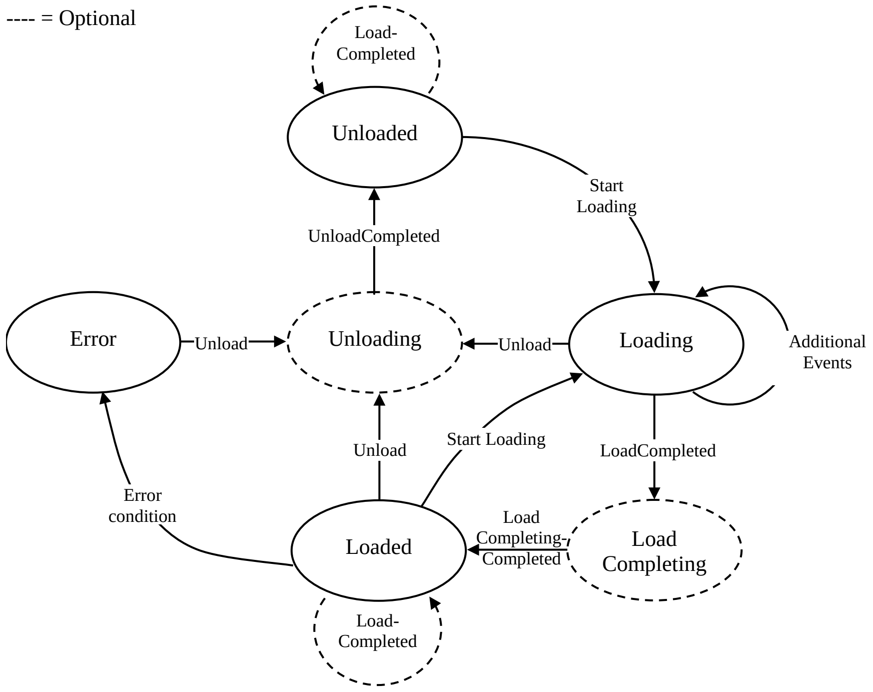


Figure 12 – Example Load State Machine

Table 59 - Load State Machine transition table

	Load State					
Load Event	**Unloaded (00h)**	**Loaded (01h)**	**Loading (02h)**	**Error (03h)**	**Unloading (04h)**	**LoadCompleting (05h)** *)
No Operation (00h)	Unloaded	Loaded	Loading	Error	Unloading	LoadCompleting
Start Loading (01h)	Loading	Loading	Loading	Error	Error	R: Error O: I: LoadCompleting →M: Loaded
Load Completed (02h)	R: Unloaded O: Error	R: Loaded O: Error	I: LoadCompleting →M: Loaded	Error	Error	R: Error O: I: LoadCompleting →M: Loaded
Additional /Segment Load Controls (03h)	R: Unloaded O: Error	Error	Loading	Error	Error	R: Error O: I: LoadCompleting →M: Loaded
Unload (04h)	Unloaded	I: Unloading →M: Unloaded	I: Unloading →M: Unloaded	I: Unloading →M: Unloaded	I: Unloading →M: Unloaded	R: I: Unloading →M: Unloaded O: I: LoadCompleting →M: Loaded
Device Restart	Unloaded	Loaded (Error in case of error detection at start-up)	R: Loading O: Error	Error	Unloaded	R: Unloaded O: I: LoadCompleting →M: Loaded

LEGEND:

I:	intermediate state
M:	mandatory
O:	optional
R:	recommended
→:	mandatory transition from a possible intermediate state to a final state
*):	Note: a device may be offline during state LoadCompleting and therefore not react to load events from a MaC. Transition from state LoadCompleting to state Loaded is an internal event. Therefore it may not be possible to provoke any state transition or a device restart from a MaC. Ignore load event: if the device may communicate, it shall return unchanged state LoadCompleting

For events that are client errors, more than one transition is allowed; the recommended transitions should be implemented but the optional transitions may be implemented alternatively. For devices with a non-loadable application the Property Load Control shall be read-only and shall have the fix value Loaded. For transitions that need more than two seconds the intermediate state is mandatory.

4.17.2.4 Usage by the Management Client

4.17.2.4.1 Sequence of downloading data via a Load State Machine

The event UnloadCompleted is mostly only an internal event. Therefore, a Management Client may never observe the state Unloading.

The complete download of loadable data controlled by a Load State Machine (e.g. Address- or Association table) shall be done as specified in Figure 13.

If a device contains more than one Load State Machine, dependencies between these Load State Machines have to be defined in the Profiles (see [15]) of these devices.

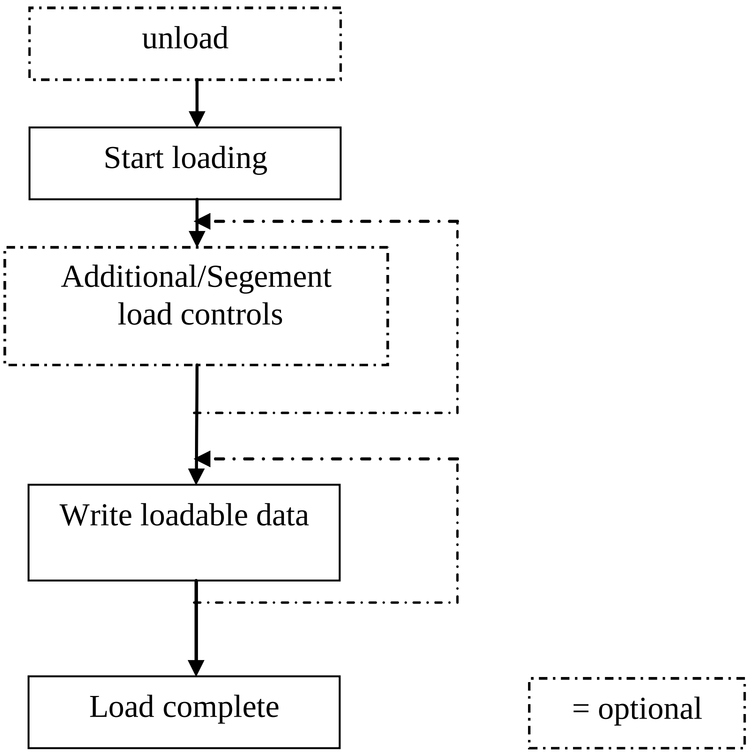


Figure 13 - Load Procedure

The event LoadCompletingCompleted is only an internal event.

If the MaC observes load state LoadCompleting, it shall

- control the load state by repeated reading of PID_LOAD_STATE_CONTROL and continue with further access only after load state has changed to LoadCompleted
- try to keep alive an established TL-connection. This is accomplished by the repeated reading of PID_LOAD_STATE_CONTROL, with a time period that is shorter than the TL-timeout. The period for reading shall not exceed half the TL-timeout, i.e. 3 seconds.

If the MaS responds during state LoadCompleting, an established TL-connection is kept alive; otherwise, the TL-connection will close after TL-timeout.

NOTE 30 A device may be offline during state LoadCompleting. A running TL-connection may be lost if the duration of state LoadCompleting exceeds the TL-timeout.

If a before established TL-connection breaks down, the MaC shall try to re-establish the connection periodically during the maximum transition time and once more when the maximum transition time has passed.

4.17.3 Load State Machine – Realisation Type 2 (Memory mapped)

This Realisation Type is not specified in this version of this document.

4.18 Run State Machine

4.18.1 Abstract Resource definition

The Run State Machine shall control the execution of an executable part of the device.

The executable part may be loaded via a Load State Machine or may be fixed in the device. If the executable part is controlled by a Management Client the Run State Machine is mandatory.

4.18.2 Run State Machine – Realisation Type 1 (Property based)

4.18.2.1 Format

Realisation Type 1 for the Run State Machine shall be a Property of Property Datatype PDT_CONTROL [84)]. The format shall differ between the read access, as specified in Table 60 and write access, as specified in Table 61.

For the minimum implementation the Run State Machine may be read-only. In this case the Run State shall show only the current state of the executable part. No external events can be transferred to the Run State Machine.

The transitions between the states shall be less than 30 seconds.

4.18.2.2 Location

This Realisation Type of the Run State Machine shall be accessed by an associated Interface Object with the Property Run Control (PID_RUN_STATE_CONTROL), see also 4.2.6.

EXAMPLE The Property Run Control (PID_RUN_STATE_CONTROL) shall be available for the Interface Object Application Program.

The implemented possible states of an executable part that is loadable via a Load State Machine are defined by the Run State Machine. The Run State shall be stored in volatile memory (e.g. RAM).

4.18.2.3 Usage by the Management Server (device)

4.18.2.3.1 States of the Run State Machine

The states of the minimum Run State Machine (data reading from the Property Run Control) shall be encoded as specified in Table 60.

Table 60 - Run States

State	Value	M/O [a]	Remarks

[84)] Existing implementations of mask 0020h and mask 0701h use PDT_UNSIGNED_CHAR. Any new implementation of these masks and any other Profile shall use PDT_CONTROL.

Halted	0	M	The executable part shall be halted or not loaded. (There shall be an automatic transition to ready if the Program is loaded.)
Running	1	M	The executable part shall be running.
Ready	2	M	The executable part shall be ready for being executed, but not yet running. If the run-conditions are fulfilled the executable part shall start automatically.
Terminated	3	O	The executable part shall be terminated and shall no longer start automatically. It shall only start again through an event “Restart” to the Run State Machine or through a reset of the device.
Starting	4	M	The executable part shall be starting and not yet ready. This state is only mandatory if an executable part needs more than 2 seconds for start-up.
Shutting down	5	M	The executable part shall be shutting down. This state is only mandatory if the executable part needs more than 2 seconds for shutting down.

[a] M: mandatory
O: optional

The state Terminated is optional.

4.18.2.3.2 Events of the Run State Machine

It shall be possible to start (if the run conditions are fulfilled) and stop the executable part for diagnostic purposes. The data for the Property Run Control shall be encoded as specified in Table 61.

Table 61 - Run State Events (Writing to Run Control)

Run State Event	Value	Remarks
NOP	0	No operation.
Restart	1	Request to restart the executable part.
Stop	2	Request to stop the executable part.

All events for the Run State Machine are optional.

4.18.2.3.3 Run state machine

Below the state transition diagram for the Run State Machine of the executable part is drawn.

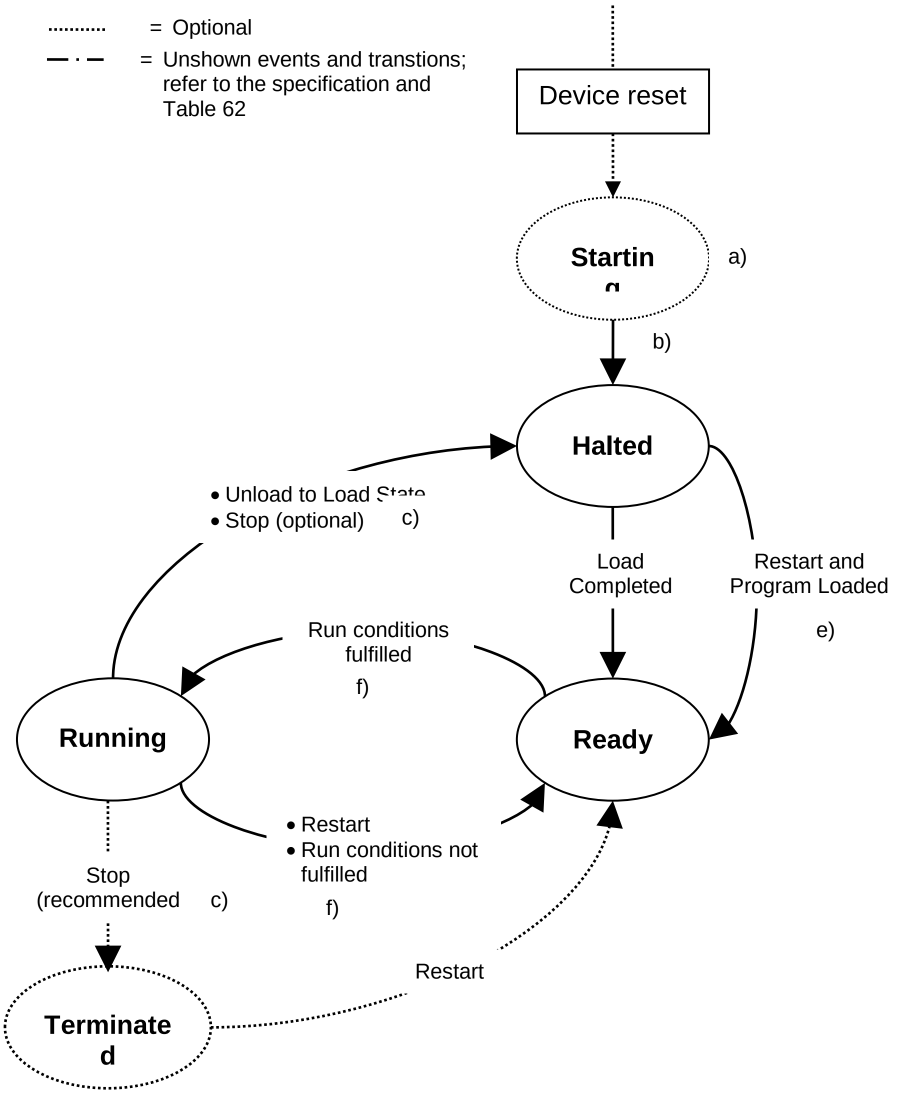


Figure 14 - Minimum Run State Machine (not all events and transitions are shown)

a) The states *Starting* and *Shutting Down* (not shown in the diagram) shall be read-only states. This means that during these states no state transition shall be done on an external event.

b) The transition from *Starting* to *Halted* shall be done internally.

The transition to *Shutting Down* shall be device - and application specific.

c) Even if not shown explicitly, the event *Stop* shall always lead to the state *Terminated*; this shall only be possible if the corresponding Load State Machine is in the state Loaded.

d) After a reset the Run State Machine shall always start in the state *Halted.*

e) Then the system shall automatically make the transition to the state *Ready* if the corresponding Load State Machine is in the state *Loaded.*

The automatic transition shall only be made at start-up. If the Run State Machine is manually stopped it shall not restart automatically.

The run state *Halted* shall only entered after start-up or if the corresponding Load State is not *Loaded*.

Starting and Shutting Down are intermediate states that may never appear.

f) The transitions between the run states *Ready* and the *Running* shall always be made automatically by the system depending on the run conditions. The run conditions are hardware specific and/or application specific.

Examples of Run conditions:

- Correct PEI-type
- Correct checksum
- Hardware check OK
- Ordernumber OK
- The Load State Machine of the corresponding application segment has the Load State equal to "Loaded".

No error state is defined for the Run State Machine. Unknown events shall be ignored.

Table 62 - Run State Machine transition table

Event	**State**					
	Halted (00h)	**Running (01h)**	**Ready (02h)**	**Terminated(03h)**	**Starting (04h)**	**Shutting down (05h)**
No Operation (00h)	Halted	Running	Ready	Terminated	I: Starting → M: Halted	I: Shutting down M: Halted
Restart and executable part loaded (01h)	I: Halted → I: Ready → M: Running	I: Halted → I: Ready → M: Running	I: Halted → I: Ready → M: Running	I: Halted → I: Ready → M: Running	I: Starting → M: Halted	I: Shutting down M: Halted
Restart and executable part unloaded (01h)	Halted	Halted	Halted	Halted	I: Starting → M: Halted	I: Shutting down M: Halted
Stop (02h)	Terminated	R: Terminated O: Halted [a]	Terminated	Terminated	I: Starting → M: Halted	I: Shutting down M: Halted
Unload to corresponding Load State	Halted	I: Shutting down M: Halted	Halted	Halted	I: Starting → M: Halted	I: Shutting down M: Halted
Device Restart and executable part loaded (PowerUp)	I: Halted → I: Ready → M: Running	I: Halted → I: Ready → M: Running	I: Halted → I: Ready → M: Running	I: Halted → I: Ready → M: Running	I: Starting → M: Halted	I: Shutting down M: Halted
Device Restart and executable part unloaded (PowerUp)	Halted	not applicable	not applicable	Halted	I: Starting → M: Halted	I: Shutting down M: Halted

LEGEND:
- I: intermediate state
- M: mandatory
- O: optional
- R: recommended

→: mandatory transition from a possible intermediate state to a final state

[a] BCU 2 goes to Terminated and masks 0300h and 2300h to Halted.

The transition from Ready to Running shall be done automatically if the run conditions are fulfilled. The state Shutting down shall either to lead an automatic restart of the device or to a power off and shall therefore always be an intermediate state.

For transitions that need more than two seconds the intermediate states Starting and Shutting down are mandatory.

Optional additional error information can be made available according the specifications of Functional Block “Technical Alarm”, see Chapter 7/1/4 “Techical Alarm”.

4.18.2.3.4 Dependencies between Load - and Run State Machine

One of the run conditions of the Run State Machine is the state of the Load State Machine. The executable part can only start if the Load State is Loaded. For an executable part that cannot be loaded the Run State may be read-only and may always be in the state Running.

4.18.2.4 Usage by the Management Client

These Run State Machine – Realisation Type 1 shall be accessed by standard Property services (A_PropertyValue_Write, A_PropertyValue_Read).

The standard error handling shall apply for the services A_PropertyValue_Read and A_PropertyValue-_Write.

Starting and Shutting Down are intermediate states that may never appear.

4.18.3 Run State Machine – Realisation Type 2 (Memory mapped)

This Realisation Type is not specified in this version of this document.

4.19 OptionReg

4.19.1 Abstract Resource definition

The OptionReg shall contain the EEPROM Option Register.

4.19.2 OptionReg – Realisation Type 1

Used by: masks 0010h, 0011h, 0012h

4.19.2.1 Format

Bits 0 and 1 of this octet are determined by the processor hardware (see datasheet).

Octet	Bit		Encoding (d = default value)
1	0	SEC (MC68HC05 specific)	
	1	EE1P (MC68HC05 specific)	0: EEPROM address range 0120h to 01FFh is protected 1: EEPROM address range 0120h to 01FFh is writeable
	2	Reserved for system software. Shall be 1	
	3	Reserved for system software. Shall be 1	
	4	Reserved for system software. Shall be 1	
	5	Reserved for system software. Shall be 1	
	6	Reserved for system software. Shall be 1	
	7	Reserved for system software. Shall be 1	

4.19.2.2 Location

Address 0100h

4.20 Programming Mode (prog_mode)

4.20.2 Programming Mode – Realisation Type 1

This is the Property PID_PROGMODE as specified in 4.3.5.

4.20.3 Programming Mode – Realisation Type 2

Used by:
- Ctrl-Mode fixed DMA
- Ctrl-Mode reloc DMA
- masks 0012h 0020h, 0021h, 0701h in E-Mode

4.20.3.1 Format and encoding

This Resource may share its storage location with other data on the same memory location inside the device. These other data may be different depending of the mask version of the device. The Resource for activation of the Programming Mode is always the same.

The Resource for activating the Programming Mode shall consist of two bits. The location inside the memory is specified in Figure 15.

curr_prog_mode							
octet 1							
7	6	5	4	3	2	1	0
p_parity	X	X	X	X	X	X	**prog_mode**

Figure 15 – Shared variable curr_prog_mode

The bit **prog_mode** (bit 0) shall serve both for setting and clearing the Programming Mode as well as for reflecting the state of the Programming Mode. If read with value “1” then the Programming Mode is active, if read with value “0” then the Programming Mode is not active. If the Programming Mode is active, then the condition for reacting on the services A_IndividualAddress_Read and A_IndividualAddress_Write (see [06]) are fulfilled.

If the Programming Mode is activated or deactivated by user interaction at the device (e.g. via programming button) this shall also reflected in this bit value. The programming mode may also be indicated at the device (e.g. programming LED) that shall then display the state of the bit **prog_mode**.

Bit 1 to bit 6 are the shared bits. They shall be don’t care in context of Programming Mode. Even if the whole octet is read and written back they shall not be changed when activating and deactivating the Programming Mode.

The variable **p_parity** shall be the parity bit of the complete octet. The don’t care bits 1 to 6 may have some essential importance for the system stack and so the complete range bit 0 to bit 6 may be parity controlled. With this also the variable **prog_mode** shall be covered by the parity check of this systems.

Even if the value of the don't care bits 1 to 6 is not evaluated, the state [85] of the variable **p_parity** shall always be changed if the variable **prog_mode** is changed. This means that if the value of **prog_mode** is changed from "0" to "1" or from "1" to "0" then the variable **p_parity** shall be inverted.

4.20.3.2 Location

The location of **curr_prog_mode** shall be the memory address 0060h.

4.20.3.3 Usage by Management Server (device)

The Management Server shall reflect the internal state of the Programming Mode to the bit **prog_mode**. If the Programming Mode is active, then the value of **prog_mode** shall be set to "1". If the Programming Mode is not active, then the value of **prog_mode** shall be set to "0". This is independent of the method by which the Programming Mode is activated or deactivated.

If **prog_mode** is set from "0" to "1" by the client then the Programming Mode shall be activated. If the **prog_mode** is set from "1" to "0" by the client then the Programming Mode shall be deactivated.

The variable **p_parity** shall always have a valid value. If no parity checking is used for the octet then the value of **p_partity** shall be ignored.

The reaction of an invalid parity value is manufacturer specific [86].

4.20.3.4 Usage by Management Client

4.20.3.4.1 General requirements

The Management Client shall set the Programming Mode via read – modify – write of the variable **curr_prog_mode**. The variable **prog_mode** shall be set to the value "1" if Programming Mode shall be activated. The variable **prog_mode** shall be set to the value "0" if Programming Mode shall be deactivated. The variable **p_parity** shall be inverted, if the value of **prog_mode** is changed.

The following Configuration Procedures specify the activation and deactivation of the Programming Mode.

4.20.3.4.2 Activation Programming Mode

Procedure: Activate prog_mode

```
    ; Connect to the device.
DMP_Connect_RCo(IA = Device_IA)
    ; Get the current programming mode of the device (curr_prog_mode).
DMP_MemRead_RCo(Addr = 0060h, length = 1)
if prog_mode = 0
    then new_prog_mode = 1
        ; Write the new programming mode.
    DMP_ProgModeSwitch_RCo(mode = new_prog_mode)
end if
    ; Disconnect from the device.
DMP_Disonnect_RCo()
```

[85] It is assumed, that a proper running system has always a valid setting of the variable p_parity.
[86] Typically the system is restarted if p_parity is invalid.

4.20.3.4.3 Deactivation Programming Mode

Procedure: Deactivate prog_mode

```
    ; Connect to the device.
DMP_Connect_RCo(IA = Device_IA)
    ; Get the current programming mode of the device curr_prog_mode.
DMP_MemRead_RCo(Addr = 0060h, length = 1)
if prog_mode = 1
    then new_prog_mode = 0
        ; Write the new programming mode.
    DMP_ProgModeSwitch_RCo(mode = new_prog_mode)
end if
    ; Disconnect from the device.
DMP_Disonnect_RCo()
```

4.20.3.4.4 Local access

The behaviour for the local access to **prog_mode** for Management Clients is the same as specified. Instead of the connection oriented procedures (RCo) the according local procedures for Emi1 or Emi2 has to be used.

4.21 Group telegram rate limitation

4.21.1 Abstract Resource definition

“Group Telegram Rate Limitation” shall be a parameter of the Application Layer. Its value shall represent the number of group telegrams that shall at maximum be sent by a device within a time period or an interframe time between two telegrams. Telegrams that are no group telegrams shall not be limited by group telegram rate limitation functionality.

4.21.3 Telegram Rate Limitation – Realization Type 2

Used by	Mask 07B0h, 17B0h

4.21.3.1 Format and location

The Telegram Rate Limitation parameter shall consist of two Properties located in the Device Object.

- PID_GROUP_TELEGR_RATE_LIMIT_TIME_BASE : see 4.3.34.
- PID_GROUP_TELEGR_RATE_LIMIT_NO_OF_TELEGR: see 4.3.34.

4.21.3.2 Discovery

Whether or not a Management Server supports Telegram Rate Limitation – Realisation Type 2 can be discovered by the Management Client by scanning these Properties in the Device Object DMP_InterfaceObjectScan_R.

5 Resources for Couplers

5.1 Resources for mask 0910h and mask 0911h

- A Couplers shall route Frames that have a Group Address as well Frames that have an Individual Address as destination address using the approach given in [04] clause 2.4.2.4.2 “Routing in case of an Individual Destination Address: Line Coupler”and clause 2.4.2.4.3 “Routing in case of an Individual Destination Address: Backbone Coupler”.
- It shall be possible to parameterize the Coupler in such a way, that independent of the communication mode and the hop counter the Coupler always generates a Layer 2 acknowledge after reception of a correct Frame.
- It shall be possible to at least route Frames with Group Addresses from Main Group 0 to 13 individually and from Main Group 14 or higher either generally or individually.
- Depending on the Individual Address, it shall be possible to use the Coupler as Line - or Backbone Coupler (optionally as KNX TP1 Repeater).
  - If used as Backbone Coupler:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Individual Address	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 to 11 = 0.

The Backbone Coupler separates the Backbone Line from the Main Line.

  - If used as Line Coupler:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Individual address	x	x	x	x	Q	Q	Q	Q	0	0	0	0	0	0	0	0

Bit 8 to 11 $\neq$ 0 and bit 0 to 7 = 0.

The Line Coupler separates the Main Line from the Line.

  - If used as a KNX TP1 Repeater:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Individual Address	x	x	x	x	Q	Q	Q	Q	D	D	D	D	D	D	D	D

Bit 8 to 11 $\neq$ 0 and bit 0 to7 $\neq$ 0.

The KNX TP1 Repeater separates physical segments in a Line.

- For bus load tests for Couplers, please refer to [03]. For corresponding tests, see [17].
- It shall be possible to reduce the number of repetitions from Primary Side to secondary Side and vice versa in case of use as a KNX TP1 Repeater for Telegrams in point-to-point as well as in point-to-multipoint communication mode.
- In case of power down of the Primary Side, it is recommended that according to parameterization the Coupler generates an error message.
- The following Profiles are defined for Couplers: Mask 0910h and mask 0911h.
  - These Profiles shall not be used for new developments of Couplers: their implementation shall be exclusively used for re-assessment of existing mask 0910h and 0912h iplementations.

– The Coupler Profiles 0910h and 0911h connect a Primary Side and a Secondary Side and contain two modified BCU 1 implementations, one per connection, and 4 kb static RAM to which both BCU 1 implementations have access. The Filter Table is stored in the upper 3,5 kb of the RAM, corresponding to 28 672 Group Addresses. The lower 0,5 kb of RAM is split into 20 Telegram buffers, 10 in either direction, with a length of 23 octets each. These buffers are used for passing Telegrams between the processors and hence to the connections (Primary Side and Secondary Side) thereby providing the routing function. The processor connected to the Primary Side is defined as the master.

▲ The Filter Table for the Masks 0910h and 0911h is organized as bit fields: If there is a Group Address represented in the Filter Table, then the corresponding bit is set to ‘1’. The method of numbering within an octet is from the lowest value to the highest value bit (from D0 to D7). If e.g. on memory address 021Dh the value 01h is written, the Group Address included in the Filter Table can then be calculated as follows:

1. The contents corresponds to 00000001b in binary form (signifying one set Group Address).
2. Up till address 021Dh, it is possible to store 1DHh x 8 = E8h Group Addresses
3. Group Address, bit position is 0 or E8h + 0 = E8h = ~~0~~*000 0***000 1110 1000** B = *0*/**232**

▲ The memory model of mask 0910h and 0911h (which shall be implemented in both processors) deviates from the BCU 1 Profile in the following way :

Address	mask 0910h	mask 0911h
109h to 10Ch	unused	unused
10Fh	identical to BCU1	unused
112h	unused	MPhyRstCnt
113h	unused	MGrRstCnt
114h	unused	SPhyRstCnt
115h	unused	SGrRstCnt
116h	shall be 1	shall be 1
119h	LKstat	LKstat
11Ah	LKerror	LKerror
11Bh	LKdef1	LKdef2
11Ch	LKgrpconf	LKgrpconf
11Dh to 1FEh	unused	unused

Possible settings of above parameters.

a) Lkdef

Setting	Meaning	mask 0910h	mask 0911h
xxxxxx11	ignore errors in group table	X	X
xxxxxx10	in case of group table errors, route all telegrams of the error group	X	X
xxxxxx01	in case of group table errors, block all telegrams of the error group	X	X
xxx1xxxx	group addresses > 6FFFH are transmitted	X	X
xxx0xxxx	group addresses >6FFFH are blocked	X	X
xx1xxxxx	group table is cyclically checked	X	X
xx0xxxxx	group table is not cyclically checked	X	X
1xxxxxxx	Main Line; routing of physical addresses activated		X
0xxxxxxx	Main Line; routing of physical addresses deactivated		X
x1xxxxxx	sub line; routing of physical addresses activated		X
x0xxxxxx	sub line; routing of physical addresses deactivated		X
xxxx1xxx	Main Line; IACK in case of illegal addressing deactivated		X
xxxx0xxx	Main Line; IACK in case of illegal addressing activated		X
xxxxx1xx	sub line; IACK in case of illegal addressing deactivated		X
xxxxx0xx	sub line; IACK in case of illegal addressing activated		X

b) Lkgrpconf

Setting	Meaning	mask 0910h	mask 0911h
xxxxxx11	group telegrams pass routed main to sub line	X	X
xxxxxx10	group telegrams blocked main to sub line	X	X
xxxxxx01	group telegrams pass not routed main to sub line	X	X
xx11xxxx	group telegrams pass routed sub line to main	X	X
xx10xxxx	group telegrams blocked sub line to main	X	X
xx01xxxx	group telegrams pass not routed sub line to main	X	X

c) Lkerror

Setting	Meaning	mask 0910h	mask 0911h
11111111	no errors	X	X
xxxxxxx0	telegram length error	X	X
xxxxxx0x	group table error	X	X
xxxxx0xx	slave error after reset	X	X
xxx0xxxx	illegal telegram in TX buffer	X	X
xx0xxxxx	connection error between two processors	X	X

d) Lkstat

Setting	Meaning	mask 0910h	mask 0911h
xxxx1110	test mode	X	X
xxxx1101	repeater	X	X
xxxx1011	line coupler	X	X
xxxx0111	bus coupling unit	X	X

e) MPhyRstCnt: number of repetitions by Line Coupler sending on the Main Line of physical addressed Telegrams after reception of NACK or BUSY. The bit configuration is shown underneath.

f) MGrRstCnt: number of repetitions by Line Coupler sending on the Main Line of group addressed Telegrams after reception of NACK or BUSY. The bit configuration is shown underneath.

g) SPhyRstCnt: number of repetitions by Line Coupler sending on the sub line of physical addressed Telegrams after reception of NACK or BUSY. The bit configuration is shown underneath.

h) SGrRstCnt: number of repetitions by Line Coupler sending on the Main Line of group addressed Telegrams after reception of NACK or BUSY. The bit configuration is shown underneath.

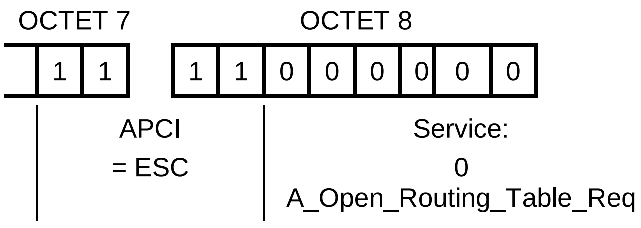
7	6	5	4	3	2	1	0
number of repetition in case of BUSY			reserved		number of repetitions in case of NACK		

- Additional Line Coupler APCIs

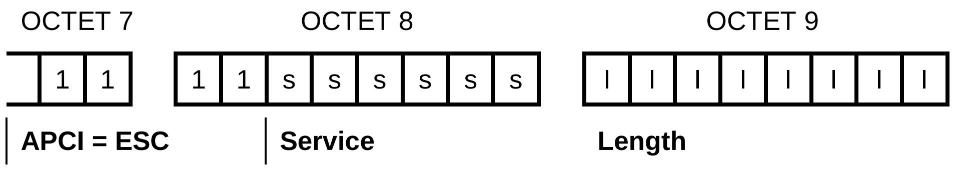
Besides the normal APCIs, which are implemented in BCU 1, a number of additional APCI's are used in profile 1.0 and 1.1 of the line coupler. These functions are available by using the APCI "ESC" (escape = 1111).

– **Enable external RAM**

Before any access to the external RAM, it has to be enabled.

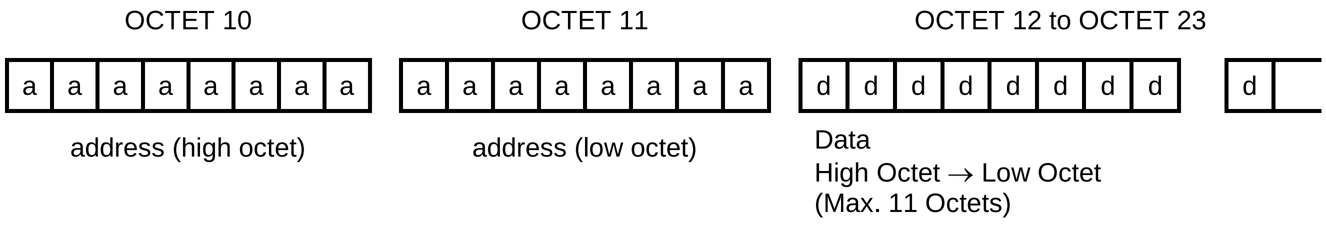


– **Read/Write external RAM/Slave**



The service shall be encoded as follows.

1h	A_Read_Routing_Table_Req
2h	A_Read_Routing_Table_Res
3h	A_Write_Routing_Table_Req
8h	A_Read_Router_Memory_Req
9h	A_Read_Router_Memory_Res
Ah	A_Write_Router_Memory_Req



– **Group Functions**

OCTET 7		OCTET 8								OCTET 9							
1	1	1	1	0	0	s	s	s	s	f	f	f	f	f	f	f	f
APCI = ESC		Service								Function							

The Service shall be encoded as follows.

Dh:	A_Read_Router_Status_Req
Eh:	A_Read_Router_Status_Res
Fh:	A_Write_Router_Status_Req

The Function shall be encoded as follows.

33h	Both Lines Normal
22h	Both Lines Block
11h	Both Lines Free
03h	Main Line to sub line Normal
02h	Main Line to sub line Block
01h	Main Line to sub line Free
30h	sub line to Main Line Normal
20h	sub line to Main Line Block
10h	sub line to Main Line Free

5.2 Resources for mask 0912h

5.2.1 Filter Table Realisation Type 2

Used by: Mask 0912h

5.2.1.1 General

Because of the memory mapped access for the Filter Table, the structure of the Filter Table shall be as implicitly fixed as specified in for masks 0910h and 0911h above. In mask 0911h Couplers the Filter Table is limited to 28 672 Group Addresses. In mask 0912h the Filter Table supports filtering of full 16 bit Group Addressing. For compatibility reasons the Filter Table shall be accessed using the already specified Coupler specific AL-services. The address 0000h shall be reserved for the broadcast address and shall not be evaluated in the Filter Table.

5.2.1.2 Location

The Filter Table Realisation Type 2 shall start at memory address 0200h and shall end on memory address 21FFh.

5.2.1.3 Format

The bit position and memory address of the octet in the Filter Table shall be related by the following.

The Filter Table of the Coupler mask 0912h shall be organized as bit fields. If there is a Group Address present in the Filter Table, the corresponding bit shall be set to ‘1’. The method of numbering within an octet shall be from the lowest value to the highest value bit (from D0 to D7).

EXAMPLE 7 If on memory address 021Dh the value 01h is written, then the Group Address included in the Filter Table can then be calculated as follows.

1. The content corresponds to 00000001b in binary form (signifying one set Group Address).
2. Up till address 021Dh, it is possible to store 1Dh x 8 = E8h Group Addresses.
3. Group Address, bit position is 0 or E8h + 0 = E8h = ~~*0*~~*000 0***000 1110 1000b** = *0*/**232.**

5.2.1.4 Usage by the Management Client and the Management Server

Octet 6								Octet 7							
						APCI									
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
						APCI	APCI	APCI	APCI						
						1	1	1	1	0	0	0	0	0	0

Figure 16 – A_Open_Routing_Table-PDU (example)

Before any access to the Filter Table it has to be enabled using this service. On reception of the A_Open_Routing_Table-PDU, the Management Server shall set the LSM of the Filter Table implemented in the Router Object to LOADED. There shall be no transition to the state ERROR due to this service or missing this service. There shall also be no dependency to the LTE Filter Table.

5.2.2 Coupler Parameters

5.2.2.1 Normal conditions

5.2.2.1.1 Introduction and overview

A Coupler according mask 0912h has two Management Servers implemented. One memory mapped based MaS for the compatibility with Coupler mask 0911h and one MaS for a Property based tool access. The Configuration Procedures for mask 0912h use a mixture of both MaS.

The memory mapped access is used for the Filter Table. Using the Function Property access can result in a faster loading of the Filter Table, but not in all cases. So it makes no sense to implement new MaC procedures for that.

For parameters the Property based access is used. The set of parameters in mask 0912h is expanded compared to the set of parameters in Couplers mask 0911h.

Mapping of Property based Parameters to memory mapped Parameters

As the memory mapped parameters are only a subset of the parameters located in Properties the mapping of Property based parameters into memory mapped parameters is not completely possible and therefore not mandatory. There is no mapping of Property based Parameters into memory mapped parameters.

Mapping of memory mapped parameters to Property based Parameters

The parameters and diagnostic data of the implementations of mask 0911h shall be memory mapped Recourses. In mask 0912h these Resources shall be implemented in Properties and for compatibility reasons also as memory mapped parameters. The synchronisation of both Resources shall ensure that memory mapped parameter settings done by ETS are available in the Property based parameters for an ETS configuration using the new Configuration Procedure. The synchronisation shall be performed after Restart or after a timeout. The timeout is triggered by write access on the memory mapped Resources. For the synchronisation a maximal timeout of 6 s after ETS configuration is allowed.

An overview to the memory mapped Resources is shown in Table 63.

Table 63 – Memory mapped Resources in masks 0911h and 0912h

Address	Mask 0911h	Mask 0912h
010Eh	RouteCnt	PID_ROUTING_COUNT in Device Object
0112h	MPhyRstCnt	PID_MAIN_LCCONFIG
0113h	MGrRstCnt	PID_MAIN_LCGRPCONFIG
0114h	SPhyRstCnt	PID_SUB_LCCONFIG
0115h	SGrRstCnt	PID_SUB_LCGRPCONFIG

Address	Mask 0911h	Mask 0912h
011Bh	LKdef2	PID_MAIN_LCGRPCONFIG PID_SUB_LCGRPCONFIG PID_MAIN_LCCONFIG PID_SUB_LCCONFIG
011Ch	LKgrpconf	PID_MAIN_LCGRPCONFIG PID_SUB_LCGRPCONFIG
011Dh to 1FEh	unused	unused

5.2.2.1.2 MPhyRstCnt

MPhyRstCnt shall contain the number of Data Link Layer repetitions that the Coupler shall perform on its Primary Side for Frames in point-to-point connectionless - or connection-oriented communication mode after the initial transmission is acknowledged with “NACK”, “no L2-ack” or BUSY. The bit configuration is shown in Figure 17.

7	6	5	4	3	2	1	0
number of repetition in case of BUSY			reserved		number of repetitions in case of NACK		

Figure 17 – Format of MPhyRstCnt

This Parameter shall be mapped to bit 2 of PID_MAIN_LCCONFIG.

Bit 2 in PID_MAIN_LCCONFIG shall be set to 0 in case “number of repetition in case of BUSY” and “number of repetitions in case of NACK” are set 0 or 1.

Optional behaviour

- Bit 2 in PID_MAIN_LCCONFIG shall be set to 0 in case of “number of repetitions in case of NACK” is set 0 or 1.
- Bit 2 in PID_MAIN_LCCONFIG shall be set to 1 in case of “number of repetition in case of BUSY” or “number of repetitions in case of NACK” is set greater than or equal to 2.

Optional behaviour

- Bit 2 in PID_MAIN_LCCONFIG shall be set to 1 in case of “number of repetitions in case of NACK” is set greater than or equal to 2.

5.2.2.1.3 MGrRstCnt

MGrRstCnt shall contain the number of Data Link Layer repetitions that the Coupler shall perform on its Primary Side for Frames sent in multicast – and broadcast communication mode after the initial transmission is acknowledged with “NACK”, “no L2-ack” or BUSY. The bit configuration is identical as in Figure 17.

This Parameter shall be mapped to bit 4 of PID_MAIN_LCCONFIG for broadcast communication mode and to bit 4 of PID_MAIN_LCGRPCONFIG for point-to-multipoint, connectionless (multicast) mode.

The mapping of this Parameter shall also reset bit 4 (BROADCAST_LOCK) in PID_MAIN_LCGRPCONFIG.

NOTE 31 This is not implemented in current implementations of mask 0912h.

Bit 4 in PID_MAIN_LCCONFIG and PID_MAIN_LCGRPCONFIG shall be set to 0 in case “number of repetition in case of BUSY” and “number of repetitions in case of NACK” are set 0 or 1.

Optional behaviour

- Bit 4 in PID_MAIN_LCCONFIG and PID_MAIN_LCGRPCONFIG shall be set to 0 in case "number of repetitions in case of NACK" is set 0 or 1.
- Bit 4 in PID_MAIN_LCCONFIG and PID_MAIN_LCGRPCONFIG shall be set to 1 in case of "number of repetition in case of BUSY" or "number of repetitions in case of NACK" is set greater than or equal to 2.

Optional behaviour

- Bit 4 in PID_MAIN_LCCONFIG and PID_MAIN_LCGRPCONFIG shall be set to 1 in case of "number of repetitions in case of NACK" is set greater than or equal to 2.

5.2.2.1.4 SPhyRstCnt

SPhyRstCnt shall contain the number of Data Link Layer repetitions that the Coupler shall perform on its Secondary Side for Frames sent in point-to-point connectionless - or connection-oriented communication mode after the initial transmission is acknowledged with "NACK", "no L2-ack" or "BUSY". The bit configuration shall be identical as in Figure 17.

This Parameter shall be mapped to bit 2 of PID_SUB_LCCONFIG.

Bit 2 in PID_SUB_LCCONFIG shall be set to 0 in case of "number of repetition in case of BUSY" and "number of repetitions in case of NACK" are set 0 or 1.

Optional behaviour

- Bit 2 in PID_SUB_LCCONFIG shall be set to 0 in case of "number of repetitions in case of NACK" is set 0 or 1.
- Bit 2 in PID_SUB_LCCONFIG shall be set to 1 in case of "number of repetition in case of BUSY" or "number of repetitions in case of NACK" is set greater than or equal to 2.

Optional behaviour

- Bit 2 in PID_SUB_LCCONFIG shall be set to 1 in case of "number of repetitions in case of NACK" is set greater than or equal to 2.

5.2.2.1.5 SGrRstCnt

SGrRstCnt shall contain the number of Data Link Layer repetitions that the Coupler shall perform on its Secondary Side for Frames sent in multicast – and broadcast communication mode after the initial transmission is acknowledged with "NACK", "no L2-ack" or BUSY. The bit configuration is shown in Figure 17.

This Parameter shall be mapped to bit 4 of PID_SUB_LCCONFIG for broadcast communication mode and PID_SUB_LCGRPCONFIG for group communication mode.

The mapping of this Parameter shall also reset bit 4 (BROADCAST_LOCK) in PID_SUB_-LCGRPCONFIG.

NOTE 32 This is not implemented in current implementations of mask 0912h.

Bit 4 in PID_SUB_LCCONFIG and PID_SUB_LCGRPCONFIG shall be set to 0 in case of "number of repetition in case of BUSY" and "number of repetitions in case of NACK" is set 0 or 1.

Optional behaviour

- Bit 4 in PID_SUB_LCCONFIG and PID_SUB_LCGRPCONFIG shall be set to 0 in case of "number of repetitions in case of NACK" is set 0 or 1.
- Bit 4 in PID_SUB_LCCONFIG and PID_SUB_LCGRPCONFIG are set to 1 in case of "number of repetition in case of BUSY" **or** "number of repetitions in case of NACK" is set greater than or equal to 2.

Optional behaviour

- Bit 4 in PID_SUB_LCCONFIG and PID_SUB_LCGRPCONFIG shall be set to 1 in case of “number of repetitions in case of NACK” is set greater than or equal to 2.

5.2.2.1.6 LKdef2

Settings of Parameters in LKdef2 shall be mapped according Table 2.

Table 2 – Mapping of Parameter LKdef2

Setting								Meaning	Profile	
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0		0911h	0912h
x	x	x	x	x	x	1	1	Ignore errors in Filter Table.	M	O
x	x	x	x	x	x	1	0	In case of Filter Table errors, route all telegrams of the error group.	M	O
x	x	x	x	x	x	0	1	In case of Filter Table errors, block all telegrams of the error group	M	O
x	x	x	1	x	x	x	x	Group Addresses > 6FFFh shall be forwarded.	M	PID_SUB_LCGRPCONFIG = XXXX01XX PID_MAIN_LCGRPCONFIG = XXXX01XX
x	x	x	0	x	x	x	x	Group Addresses >6FFFh shall be blocked	M	PID_SUB_LCGRPCONFIG = XXXX10XX PID_MAIN_LCGRPCONFIG = XXXX10XX
x	x	1	x	x	x	x	x	The Filter Table shall be cyclically checked.	M	O
x	x	0	x	x	x	x	x	The Filter Table shall not be checked cyclically.	M	O
1	x	x	x	x	x	x	x	The routing of telegrams in point-to-point connectionless - and connection-oriented communication mode from the Primary Side to the Secondary Side shall be enabled.	M	PID_MAIN_LCCONFIG = XXXXXX11
0	x	x	x	x	x	x	x	All frames in point-to-point connectionless - and connection-oriented communication mode from the Primary Side to the Secondary Side shall be routed independent of the Coupler Mode	M	PID_MAIN_LCCONFIG = XXXXXX01
x	1	x	x	x	x	x	x	The routing of telegrams in point-to-point connectionless - and connection-oriented communication mode from the Secondary Side to the Primary Side shall be enabled.	M	PID_SUB_LCCONFIG = XXXXXX11
x	0	x	x	x	x	x	x	All frames in point-to-point connectionless - and connection-oriented communication mode from the Secondary Side to the Primary Side shall be routed independent of the Coupler Mode	M	PID_SUB_LCCONFIG = XXXXXX01
x	x	x	x	1	x	x	x	Primary Side: send Layer 2 acknowledge only in case of routing	M	PID_MAIN_LCCONFIG = 011XXXXX
x	x	x	x	0	x	x	x	Primary Side: always send Layer 2 acknowledge independent of routing	M	PID_MAIN_LCCONFIG = 100XXXXX
x	x	x	x	x	1	x	x	Secondary Side: send Layer 2 acknowledge only in case of routing	M	PID_SUB_LCCONFIG = 011XXXXX
x	x	x	x	x	0	x	x	Secondary Side: always send Layer 2 acknwoledge independent of routing	M	PID_SUB_LCCONFIG = 100XXXXX

5.2.2.1.7 LKgrpconf

Settings of Parameters in LKgrpconf shall be mapped according Table 3.

Table 3 – Mapping of Parameter LKgrpconf

Setting								Meaning	Profile	
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0		0911h	0912h
x	x	x	x	x	x	1	1	Telegrams in multicast communication mode from Primary Side to Secondary Side shall be routed in function of the evaluation of the Filter Table.	M	PID_MAIN_LCGRPCONFIG = XXXXXX11
x	x	x	x	x	x	1	0	Telegrams in multicast communication mode from Primary Side to Secondary Side shall be blocked.	M	PID_MAIN_LCGRPCONFIG = XXXXXX10
x	x	x	x	x	x	0	1	Telegrams in multicast communication mode from Primary Side to Secondary Side shall all be routed without evaluation of the Filter Table.	M	PID_MAIN_LCGRPCONFIG = XXXXXX01
x	x	1	1	x	x	x	x	Telegrams in multicast communication mode from Secondary Side to Primary Side shall be routed in function of the evaluation of the Filter Table.	M	PID_SUB_LCGRPCONFIG = XXXXXX11
x	x	1	0	x	x	x	x	Telegrams in multicast communication mode from Secondary Side to Primary Side shall be blocked.	M	PID_SUB_LCGRPCONFIG = XXXXXX10
x	x	0	1	x	x	x	x	Telegrams in multicast communication mode from Secondary Side to Primary Side shall all be routed without evaluation of the Filter Table.	M	PID_SUB_LCGRPCONFIG = XXXXXX01

5.2.2.2 Error and exception handling

The parameters of Couplers mask 0911h shall be a subset of the mask 0912h parameters. The Configuration Procedure ensures the possibility for mixing in the configuration of a mask 0912h Coupler the Configuration Procedures of mask 0911h (memory mapped) with the Configuration Procedures of mask 0912h (Property Based). For further details about the Configuration Procedures, please refer to [09].

5.2.3 Load State Machine – Realisation Type 1 (Property based)

The Load State Machine transition table allows an optional transition from state “Loaded” to “Error” in case of an event “Load Completed”. This is not allowed for mask version 0912h Couplers. Mask 0912h shall stay in state “Loaded” in case of an error.

Appendix A

(informative)

Examples of table implementations

A.1 Introduction

This annex gives informative examples of how the Resources may be implemented on KNX RF. For KNX RF, these Resources are not standardised and not mandatory. The configuration of KNX RF devices uses dedicated services in function of the Configuration Mode, as specified in [09]. Also the S-Mode access to KNX RF devices is specified without requiring certain formats of the Resources.

A.2 Extension of KNX to enable use of Extended Addresses

A.2.1 Existing resources from KNX

A.2.1.1 Object table (Datapoint Table)

The list of Datapoints used by a device is defined in the Datapoint Table.

Datapoint#	Datapoint Table	
	Length	(1octet)
0	Datapoint description #0	(3 octets)
1	Datapoint Description #1	(3 octets)
2	Datapoint description #2	(3 octets)
3	…..	…..

Each Datapoint is identified by an index called Datapoint#.

Property access:

Group object table Interface Object / PID_TABLE = 23 Group Object Table

A.2.1.2Address Table

Address#	Address Table	
	Length	(1octet)
0	Individual Address	(2 octets)
1	Group Address = @bank + @link	(2 octets)
2	Group Address = @bank + @link	(2 octets)
3	…..	…..

Each address is identified by an index called address# . There is no need to make the separation into @bank and @link.

Property access:

Address Table Interface Object / PID_TABLE = 23 Address table Property format 0

A.2.2 New resources

The list of Extended Group Address used in a device is defined by two tables:

- Serial number table, and
- an Association Table to link Group Addresses and Serial Number to build Extended Addresses.

A.2.2.1Serial number table

serial#	Serial number table	
	Length	(1octet)
0	Serial N° 1	(6 octets)
1	Sérial N° 2	(6 octets)
2	Sérial N° 3	(6 octets)
3	…..	…..

Each Serial Number is identified by a index called serial#.

A.2.2.1.1 New Association Table

To enable that a Datapoint can be part of multiple groups (e.g. multiple push-buttons command one lamp, one for on/off and a second for general off), an Association Table is used. This assocition links one Datapoint with one or more Extended Group Address (couple of Address and Serial Number).

Each association is defined by

- Address index (1octet),
- Serial Number index (1 octet),
- Datapoint index (1 octet).

Association table			
Length			(1octet)
Address #	Serial #	Datapoint#	(3 octets)

If a Datapoint is transmit only, the association Serial Number index equals FFh. No Serial Number can be associated.

Rules

The association of the sending Datapoint must be at the beginning of the Association Table.

The Serial Number Table Interface Object

The Serial Number Table Object includes Serial Numbers of devices linked to the device. It provides the management operations for downloading.

Table 64 - Serial number Interface Object

Property Name	Property ID	Type	Optional / Mandatory / Writable	
Object Type	PID_OBJECT_TYPE	PDT_UNSIGNED_INT	M	Serial Number tablec Object object_type 5dec
Object Name	PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[]	O	Name of the Address table
Load Control	PID_LOAD_STATE_CONTROL	PDT_CONTROL	M / W	for further Information see Load / State machines
Serial Number Table	PID_TABLE(23)	PDT_UNSIGNED_INT[]	M / W	Serial number table
..				

The Serial Number Table Property contains only Serial Numbers used by the device. If the current length is 0 all Serial Numbers are accepted. To accept no Serial Number set the load control to unloaded. The current length is the number of entries in the Address Table. The Maximum Length is transferred with the load control MAX_LENGTH or is fixed in the device.

Theoretical Serial NumberTable Format :

serial#	Serial number table	
	Nb of Serial Number	(1octet)
0	Serial N° 1	(6 octets)
1	Sérial N° 2	(6 octets)
2	Sérial N° 3	(6 octets)
3	…..	…..

A.3 Association Table Object

A.3.1 Existing object

The Association Table Object is used to connect Group Objects to Group Addresses. It provides the management operations for downloading.

Table 65 - Association table Interface Object

Property Name	Property ID	Type	Optional / Mandatory/ Writable	
Object Type	PID_OBJECT_TYPE	PDT_UNSIGNED_INT	M	Associationtable Object 2 dec
Object Name	PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[]	O	Name of the Association table
Association table Pointer	PID_TABLE_Refence	PDT_GENERIC_03	O	Pointer to Association table and maximum entries
Load Control	PID_LOAD_STATE_CONTROL	PDT_CONTROL	M / W	for further Information see Load / State- machines
Association table	PID_TABLE (23)	PDT_GENERIC_02	M / W	
..				

The Association Table Property contains associations between the communication number and the Service Access Point (SAP). The maximum number of elements in the Property shows the maximum size of the table and current length shows the current usage. In each association the high octet is the communication number and the low octet is the SAP number.

In case of sending the first entry in the Association Table shall be used.

If the current length is set to 0 a standard a Association Table shall be used (communication number + 1 = SAP number). The Maximum Length shall be transferred with the load control MAX_LENGTH or shall be fixed in the device.

A.3.2 New Property : Association Table format 1

To enable use of Extended Group Addresses, the association must be made between a Serial Number, a Group Address and a Group Object. For this a new format is designed for the Association Table Object.

The existing one just changes its Property name from Association_Table to Association_Table_format0. The new one gets the Property name Associtaion_Table_Format1.

Property Name	Property ID	Type	Optional / Mandatory/ Writable	
Object Type	PID_OBJECT_TYPE	PDT_UNSIGNED_INT	M	Associationtable Object 2 dec
Object Name	PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[]	O	Name of the Association table
Association table Pointer	PID_TABLE_REFERENCE	PDT_GENERIC_03	O	Pointer to Association table and maximum entries
Load Control	PID_LOAD_STATE_CONTROL	PDT_CONTROL	M / W	for further Information see Load / State-machines
Association table Format 0	PID_TABLE (23)	PDT_GENERIC_02	M / W	
Association table format 1	PID_TABLE (52)	PDT_GENERIC_03	O	Association table for extended group adresses

Table 66 - Association table Interface Object

In the Association table format 1, one entry consist of an association of a Serial Number index, a communication number and the Service Access Point (SAP). The maximum number of elements in the Property shows the maximum size of the table and current length shows the current usage. In each association, the octet 2 is Group Address index, the octet 1 is the Serial Number index and the octet 0 is the Group Object index.

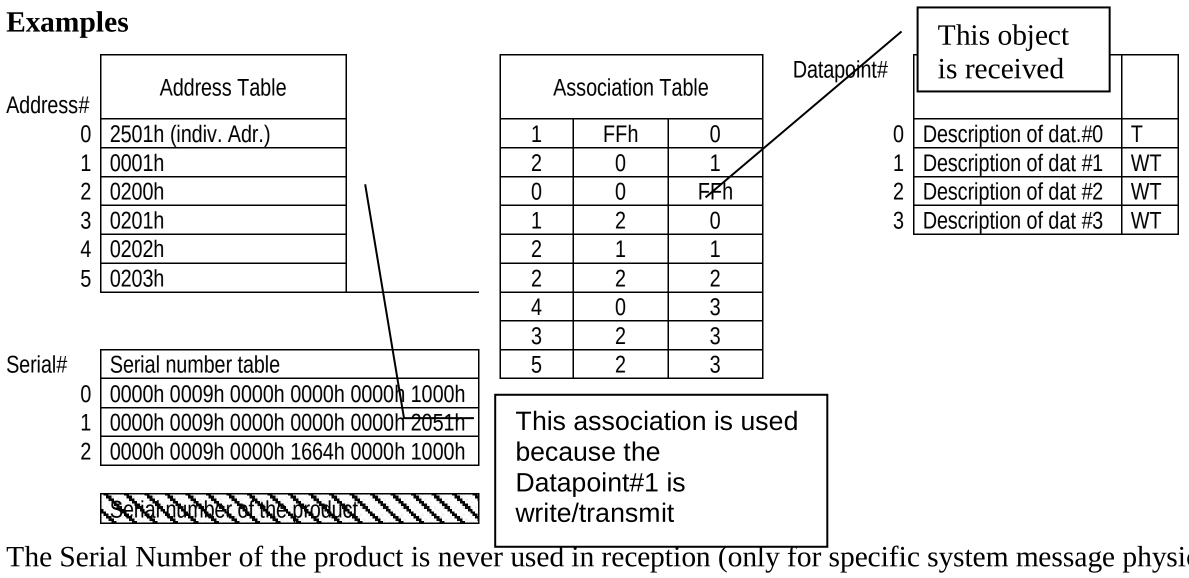
In case of sending the first entry in the association table is used.

For Association for extended Individual Addresses have the datapoint index equal to FFh.

A.4 Examples of usage

A.4.1 Use of tables in case of reception

Examples



Address#	Address Table
0	2501h (indiv. Adr.)
1	0001h
2	0200h
3	0201h
4	0202h
5	0203h

Serial#	Serial number table
0	0000h 0009h 0000h 0000h 0000h 1000h
1	0000h 0009h 0000h 0000h 0000h 2051h
2	0000h 0009h 0000h 1664h 0000h 1000h

▨ Serial number of the product

Association Table		
1	FFh	0
2	0	1
0	0	FFh
1	2	0
2	1	1
2	2	2
4	0	3
3	2	3
5	2	3

Datapoint#		
0	Description of dat.#0	T
1	Description of dat #1	WT
2	Description of dat #2	WT
3	Description of dat #3	WT

This object is received
This association is used because the Datapoint#1 is write/transmit

The Serial Number of the product is never used in reception (only for specific system message physical addressing using Serial number).

If a message is received with extended Individual Address (2501h,0000h 0009h 0000h 0000h 0000h 1000h), it is accepted (Serial Number present in the table).

If a message is received with extended Individual Address (2501h,0000h 0009h 0000h 0000h 0000h 1001h), it is ignored (Serial Number not present in the table).

If a message is received with Extended Group Address (0200h,0000h 0009h 0000h 0000h 0000h 1000h) the value of data is copied in the Datapoint# 1.

If a message is received with Extended Group Address (0200h,0000h 0009h 0000h 1664h 0000h 1000h) the value of data is copied in the Datapoint# 2.

If a message is received with Extended Group Address (0001h,0000h 0009h 0000h 0000h 0000h 1000h) it is ignored.

If a message is received with Extended Group Address (0201h,0000h 0009h 0000h 0000h 0000h 1000h) it is ignored.

A.4.2 Use of tables for sending

In case of sending the Serial Number used is always the one of the product, and not the one from the Extended Address associated to the Datapoint. The column of serial index (serial#) is not used in sending. The Serial Number of the product is stored in an other place of the memory (it is not easy to implement a length variable table with a Serial Number factory written). The first association found in the table is used for the transmission.

Address #	Address Table
0	2501h (Indiv. Addr.)
1	0001h
2	0200h
3	0201h
4	0202h
5	0203h

Association Table		
1	FFh	0
2	0	1
0	0	FFh
1	2	0
2	1	1
2	2	2
4	0	3
3	2	3
5	2	3

Datapoint#	Datapoint Table	
0	Description of dat.#0	T
1	Description of dat #1	WT
2	Description of dat #2	WT
3	Description of dat #3	WT

Serial#	Serial Number table
0	0000h 0009h 0000h 0000h 0000h 1000h
1	0000h 0009h 0000h 0000h 0000h 2051h
2	0000h 0009h 0000h 1664h 0000h 1000h

Serial number of the product