System Specifications

3

Management

5

Management Procedures

2

Summary

The Management Procedures describe the common set of procedures for Network - and Device Management shared by and independent of the Configuration Procedures.

Version 01.09.02 is a KNX Approved Standard.

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

Document Updates

Version	Date	Modifications
1.2	2001.12.20	Preparation of the Approved Standard v1.2.
1.3	2003.11.25	Integrated from S22:
	2005.08.22	Inclusion of 3.2.7 “Procedure: DM_DeviceDescriptor_InfoReport” from AN045.
	2005.09.09	From S06 “Individual Address Assignment by Serial Number” S06 clause 2.1 → 2.21 S06 clause 2.3 → 2.5 S06 clause 2.4 → 2.4
	2005.11.02	Editorial correction of Verify Mode control: is not in bit 4 of PID_DEVICE_CONTROL but in bit 2.
	2006.02.20	Integration of NM_NetworkParameter_Read
	2006.04.07	Integrated Managemen Procedures of S04 “Reduced Interface Objects”. See clause 3.22.3, 3.24.3 and 3.25.3.
	2006.07.27	Integration of the Load Event Relative Allocation
	2006.10.02	Integration of AN089: DM_Identify_RCo2
	2007.01.19	• S08 “Distributed Address Assignment” -
	2007.05.09	Added NM_SubnetworkAddress_Write Restructured first part on Network Managament (first all DoA, then all IA, then all DA).
	2007.05.14	• S09 “Subnetwork Address Management” Added NM_IndividualAddress_SerialNumber_Report
	2007.08.21	• AN092 “A_PropertyDescription_Response-PDU”
	2007.08.28	• AN026 “Use of Domain Address on RF” NM_IndividualAddress_Write2 replaced by NM_DomainAddressAndIndividualAddress_Write2. New: DM_FunctionProperty_Write_R
	2007.09.18	• AN030 “SNA Read from Router”
	2007.09.19	• AN034 “Connectionless A_Restart”
	2007.09.20	• AN065 “Number field in memory services” added the indication
	2007.11.06	• Introduction of general procedure NM_NetworkParameter_Write_R. See 2.19.1.
	2007.11.26	• Replaced PEI_Memory_Write by PC_SET_VAL • Inserted clause
	2008.01.14	• DM_DomainAndIndividualAddress_Write3
	2008.01.17	• AN044 “RF Specification Complements” extended NM_DomainAndIndividualAddress_Write2 with A_Restart.
	2008.05.06	• S15 “Easy Common Parts” AN071 “DMA and Link Services: see 3.34
	2008.05.07	• AN072 “A_Authorize” integrated.
	2008.05.14	• Editorial review. • AN046 “Device Descriptor error handling” integrated.
	2008.05.20	• AN057 “System B” added the Load Event 03h Subtype 0Bh “Data Relative Allocation”: see clause 3.28.3.4
	2008.08.01	• AN081 “2-level topology” integrated.
	2008.08.04	• Moved NM_SubnetworkAddress_Read_R to Chapter 3/5/3 “Configuration Procedures”
	2008.08.12	• AN103 “Structure of PID_MGT_DESCRIPTOR_01” integrated.
1.4	2009.01.09	• Preparation of the Approved Standard v1.4
1.4.01 AS	2009.10.13	Editorial update
1.5.00	2009.11.09	• AN118 “cEMI Transport Layer” integrated.
1.5.01	2010.10.22	• AN127 “Master Reset” integrated.
	2011.01.05	• AN124 “Interface Object index Discovery” integrated.

Version	Date	Modifications
01.06.00	2012.09.11	• Revision of the integration of AN127 “Master Reset” according the updated AN127 “Master Reset” v05.
01.07.00	2013.07.17	• AN139 "Procedures for the assignment of IAs to KNXnet/IP Tunnelling connections" integrated
01.07.01	2013.07.17	• Editorial review.
01.08.00	2013.07.22	• AN132 "A_NetworkParameter_InfoReport" integrated.
01.08.01	2013.09.10	• Editorial review.
01.09.00	2013.10.23	• AN162 "System aspects of RF S-Mode" integrated.
01.09.01	2013.10.29	Editorial updates for the publication of KNX Specifications 2.1.
01.09.02	2013.11.29	Editorial updates.

References

[01] Chapter 3/3/4 “Transport Layer”
[02] Chapter 3/3/7 “Application Layer”
[03] Chapter 3/5/1 “Resources”
[04] Chapter 3/5/3 “Configuration Procedures”
[05] Chapter 3/6/3 “External Message Interface”
[06] Chapter 3/7/2 “Datapoint Types”
[07] Chapter 3/8/1 “KNXnet/IR Remote Diagnosis and Configuration”
[08] Volume 6 “Profiles”
[09] Part 9/4 “BCUs and BIMs”
[10] AN160 “RF S-Mode Device Profiles”
[11] AN161 “Coupler Model 2.0”

Filename: 03_05_02 Management Procedures v01.09.02 AS.docx
Version: 01.09.02
Status: Approved Standard
Savedate: 2013.12.10
Number of pages: 159

Contents

1 Introduction

1.1 Definitions

The Management Procedures capture the dynamics of managing distributed Resources on the network in terms of abstract procedures. On the network itself, a procedure consists of a sequence of telegrams, exchanged between two partners: the Management Client and the Management Server.

The Management Client is a powerful device with 'controller' functionality, typically but not exclusively PC-based. Except for Network Management, the Management Server is always one particular 'target device'. In the former case, it is in fact the network as a whole that acts as partner or server. Ultimately of course, the response to a Management Client request is always generated by the individual devices connected to the network, either one or many. In addition to its runtime behaviour (based on group communication), every device moreover supports a rich Management Server profile to this purpose. One important objective of this Chapter “Management Procedures” is precisely to allow a concise description of such a Profile (see [08]). It is clear that the information about the full set of Management Procedures supported by a particular device or implementation, tells us significantly more about the device than merely the list of services through which this is realised.

Note that in general, one single device may well implement both Management Client - as well as Management Server functionality. For and during the execution of one particular Management Procedure however, one device takes on one single role.

Network Management Procedures and Device Management Procedures

So, in conclusion, there are two main classes of Management Procedures.

1. **Network Management Procedures** describe the device independent Management Procedures on the network. These are e.g. reading / writing the Individual Address, scanning the network. For these procedures no knowledge of the single devices is required.
2. **Device Management Procedures** describe the Management Procedures to access one specific device. These Management Procedures describe e.g. the load procedures, reading the state. For these Management Procedures a detailed knowledge of the device is required.

1.2 Naming conventions

The majority of the common, runtime communication on KNX bases on point-to-multipoint, connectionless communication mode (multicast).

- ***Remote Management*** denotes the management of a device via the bus. For this, the Management Procedures mainly base on the other communication modes as specified in the Transport Layer specifications in [01]. The main communication mode used in a Management Procedure is reflected as a suffix in the name of the Management Procedure, as indicated in Table 1.
- ***Local Management*** denotes the management of a device via its possible EMI-interface (EMI1, EMI2 or cEMI) as specified in [05]. This makes use of the local services, as provided by the device’s EMI-flavour, which is again indicated by a suffix in the name of the Management Procedure, as indicated in Table 1.

Table 1 – Indications in names of Management Procedures

Suffix	Communication Mode
RCo	Remote Management, using point-to-point connection-oriented communication mode
RCoV	Remote Management, using point-to-point connection-oriented communication mode. Additionally, the Verify Mode is used. NOTE Verify Mode is specified in [03].
RCl	Remote Management, using point-to-point connectionless communication mode
LEmi1	Local Management using EMI1.
LEmi2	Local Management using EMI2.
LcEmi	Local Management using cEMI.

1.3 Timing aspects of Management Procedures

The timings in this document are in certain cases multiplied by a medium dependent factor T_{media}. This factor is specified in Table 2.

Table 2 – Overview T_{media}

Medium	T_{media}
TP1	20 ms
PL110	390 ms

2 Network Management Procedures

2.1 Scope

The Network Management Procedures describe the device independent Management Procedures. These procedures are used to configure the network, and to get the information about the configuration of the network and connected devices.

For these procedures no knowledge of the single devices is required. They will work with every device connected to the network [1)].

These Management Procedures work independent of the location of the Management Client in the network. Some of these Network Management Procedures require the preceding configuration of Routers and Domain Addresses via other Management Procedures.

2.2 NM_IndividualAddress_Read

Use

This Network Management Procedure shall be used to read out the Individual Addresses of all the devices that are in Programming Mode.

This procedure shall work independently of the configuration of the Individual Address of the Routers. When applicable this procedure shall be preceded by the configuration of the Domain Address.

Used Application Layer Services for Management

- A_IndividualAddress_Read

Parameters of the Management Procedure

NM_IndividualAddress_Read(/* [out] */ individual_addresses[])

individual_addresses[]: The collection of all the Individual Addresses of the devices that are in Programming Mode.

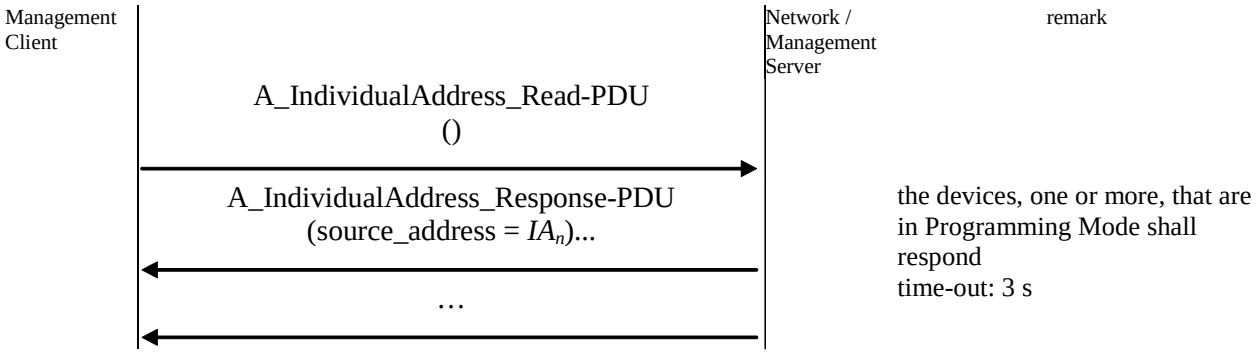
Service parameters

None.

Variables

$IA_{n:}$ The IA of one device n that responds to the A_IndividualAddress_Read-PDU. The Management Client shall collect all IA_n of the individual responses and report these via individual_addresses[].

Sequence



1) The Management Server functionality has to be implemented.

Exception handling

The Management Client shall always wait until the time-out has elapsed. It shall collect all responses IA_n during this time-out.

- If no A_IndividualAddress_Response-PDU is received, no device is in Programming Mode.
- If one A_IndividualAddress_Response-PDU is received, exactly one device is in Programming Mode.
- If more than one response is received, several devices are in Programming Mode.
- If two or more responses with the same Individual Address are received, there is more than one device with the same Individual Addresses.

The Management Client shall not evaluate Layer-2 repetitions.

2.3 NM_IndividualAddress_Write

Use

This Network Management Procedure shall be used to write the Individual Address of one single device that is in Programming Mode.

The procedure shall wait until exactly one device is in Programming Mode. It shall check that no other device has the same Individual Address. The procedure shall check if the programming is successful and shall deactivate the Programming Mode by executing a restart of the device.

When applicable this procedure shall be preceded by the configuration of the Individual Addresses of the installed Routers and the Domain Addresses.

Used Application Layer Services for Management

- A_IndividualAddress_Read
- A_IndividualAddress_Write
- A_DeviceDescriptor_Read
- A_Restart
- A_Connect

Parameters of the Management Procedure

NM_IndividualAddress_Write(/* [in] */ IA_new)

IA_new: The new IA that shall be assigned to the device in Programming Mode.

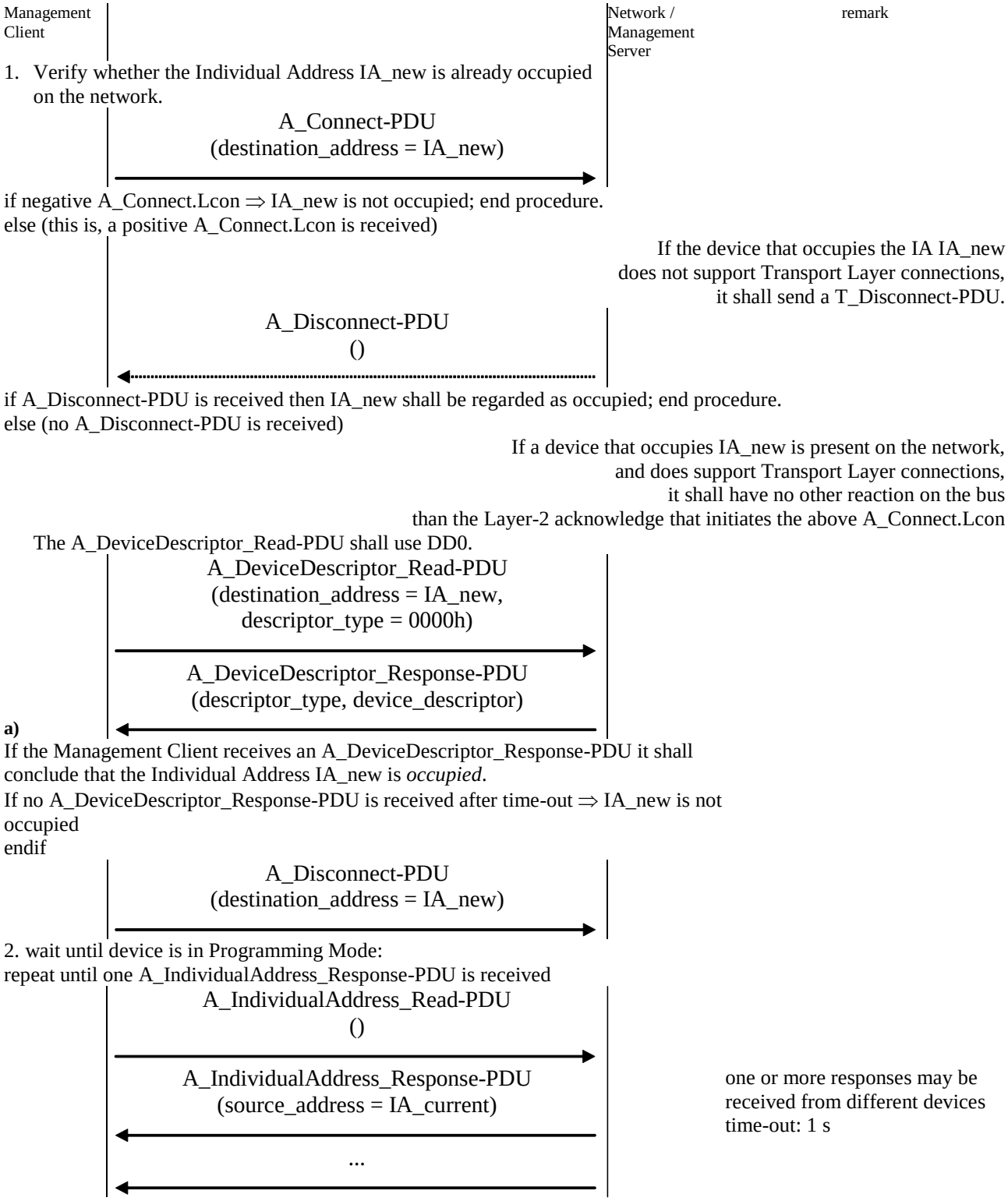
Service parameters

None.

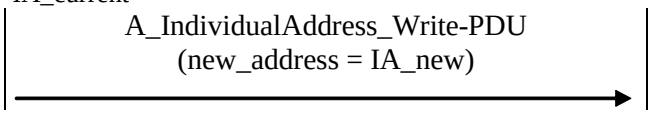
Variables

IA_current: The current IA of the device that is in Programming Mode prior to the assignment of IA_new.

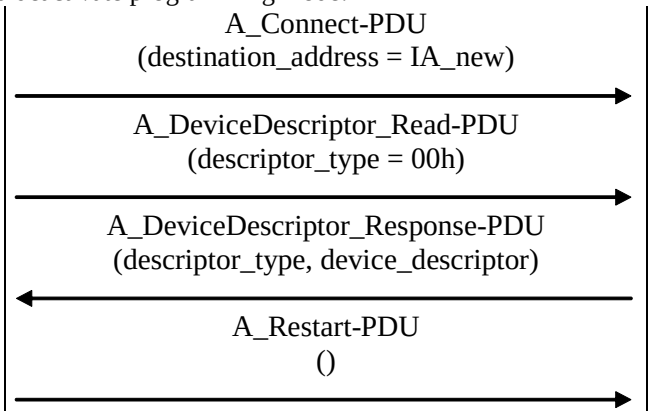
Sequence


Management Client
Network / Management Server
remark
1. Verify whether the Individual Address IA_new is already occupied on the network.
A_Connect-PDU
(destination_address = IA_new)
if negative A_Connect.Lcon ⇒ IA_new is not occupied; end procedure.
else (this is, a positive A_Connect.Lcon is received)
If the device that occupies the IA IA_new does not support Transport Layer connections, it shall send a T_Disconnect-PDU.
A_Disconnect-PDU
()
if A_Disconnect-PDU is received then IA_new shall be regarded as occupied; end procedure.
else (no A_Disconnect-PDU is received)
If a device that occupies IA_new is present on the network, and does support Transport Layer connections, it shall have no other reaction on the bus than the Layer-2 acknowledge that initiates the above A_Connect.Lcon
The A_DeviceDescriptor_Read-PDU shall use DD0.
A_DeviceDescriptor_Read-PDU
(destination_address = IA_new,
descriptor_type = 0000h)
A_DeviceDescriptor_Response-PDU
(descriptor_type, device_descriptor)
a)
If the Management Client receives an A_DeviceDescriptor_Response-PDU it shall conclude that the Individual Address IA_new is *occupied*.
If no A_DeviceDescriptor_Response-PDU is received after time-out ⇒ IA_new is not occupied
endif
A_Disconnect-PDU
(destination_address = IA_new)
2. wait until device is in Programming Mode:
repeat until one A_IndividualAddress_Response-PDU is received
A_IndividualAddress_Read-PDU
()
A_IndividualAddress_Response-PDU
(source_address = IA_current)
one or more responses may be received from different devices
time-out: 1 s
...
if more than one response is received ⇒ more than one device in Programming Mode
end repeat

3.set Individual Address
if IA_new!= IA_current



endif
4. verify and deactivate programming mode:



b)

Abort the connection of the client side Transport Layer.

Exception handling

to 1.: If an A_Disconnect-PDU is received instead of an A_DeviceDescriptor_Response-PDU, than a device with this Individual Address exists but it may either already have another Transport Layer connection open and not accept any further Transport Layer connections, or does not support connection oriented communication mode.
The Management Client shall continue with the Management Procedure in every case.

a) The Management Client shall accept any value of descriptor_type, also values ≠ 0, and any value of device_descriptor.

to 2.: The Management Client shall always wait until the time-out has elapsed. It shall collect all the responses during this time-out.
This procedure shall wait until exactly one device is in Programming Mode [2)].

Following case may occur at this point:

- A device with the Individual Address IA_new to be assigned exists, but it is <u>not</u> the one that is in Programming Mode.
  ⇒ The Management Client shall not continue with the Management Procedure.
- A device with the Individual Address IA_new to be assigned exists, and it is the one that is in Programming Mode.
  ⇒ The Management Client shall continue with the Management Procedure.
- No device with the Individual Address IA_new to be assigned exists.
  ⇒ The Management Client shall continue with the Management Procedure.

to 4.: If no A_DeviceDescriptor_Response-PDU is received, than the programming of the Individual Address may have failed, or the system (Router) is not configured correctly.

[2)] The user of the Management Client should get an information in how many devices are Programming Mode is active (none or more than one).

2.4 NM_IndividualAddress_SerialNumber_Read

Use

This Network Management Procedure shall be used to read the Individual Address of one single device of which the KNX Serial Number is known.

The KNX Serial Number of the device (SN_Device) must be known in advance.

If the Individual Address of more than one device has to be read, this Network Management Procedure "NM_IndividualAddress_SerialNumber_Read" has to be repeated for each device, using each device's KNX Serial Number.

Used Application Layer Services for Management

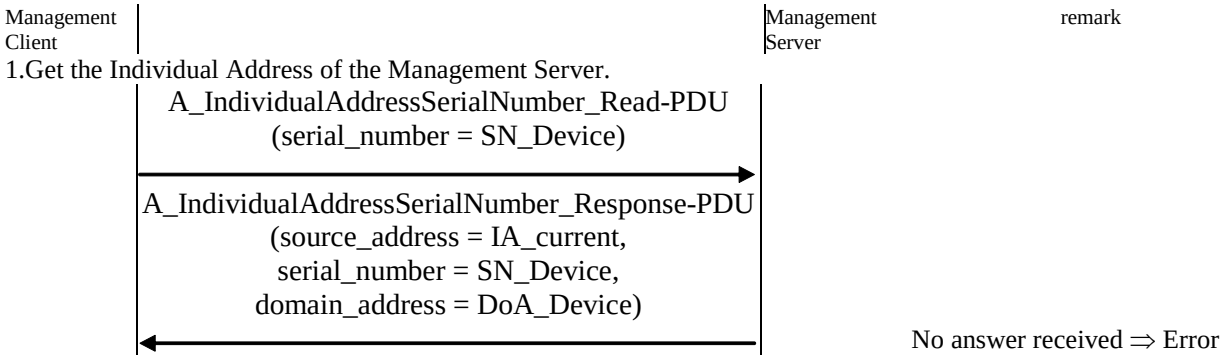
- A_IndividualAddressSerialNumber_Read

Parameters of the Management Procedure

NM_IndividualAddress_SerialNumber_Read(/* [in] */ SN_Device, /* [out] */ DoA_current, /* [out] */ IA_current)

SN_Device:	KNX Serial Number of the device of which the Individual Address is to be read.
DoA_Device:	The Domain Address of the device of which the Individual Address is read; it is contained in the response if the device is on Powerline.
IA_Device:	The Individual Address of the device, in the response.

Sequence



The Individual Address is contained as the Source Address of the A_IndividualAddressSerialNumber_Response-PDU.

Exception handling

If no answer is received, there is no device present in the network with the given KNX Serial Number.

2.5 NM_IndividualAddress_SerialNumber_Write

Use

This Network Management Procedure shall be used to write the Individual Address of one single device of which the KNX Serial Number is known.

The procedure shall ensure that the assigned Individual Address is unique. The procedure shall check if the programming has been successful.

If applicable this procedure shall be preceded by the configuration of the Individual Addresses of the installed Routers and the Domain Addresses.

The KNX Serial Number of the device to be programmed must be known in advance. Either by the mechanism NM_SerialNumberDefaultIA_Scan or by any other means.

If the Individual Address of more than one device has to be programmed, this Network Management Procedure NM_IndividualAddress_SerialNumber_Write has to be repeated for each deice, using that device's KNX Serial Number.

Used Application Layer Services for Management

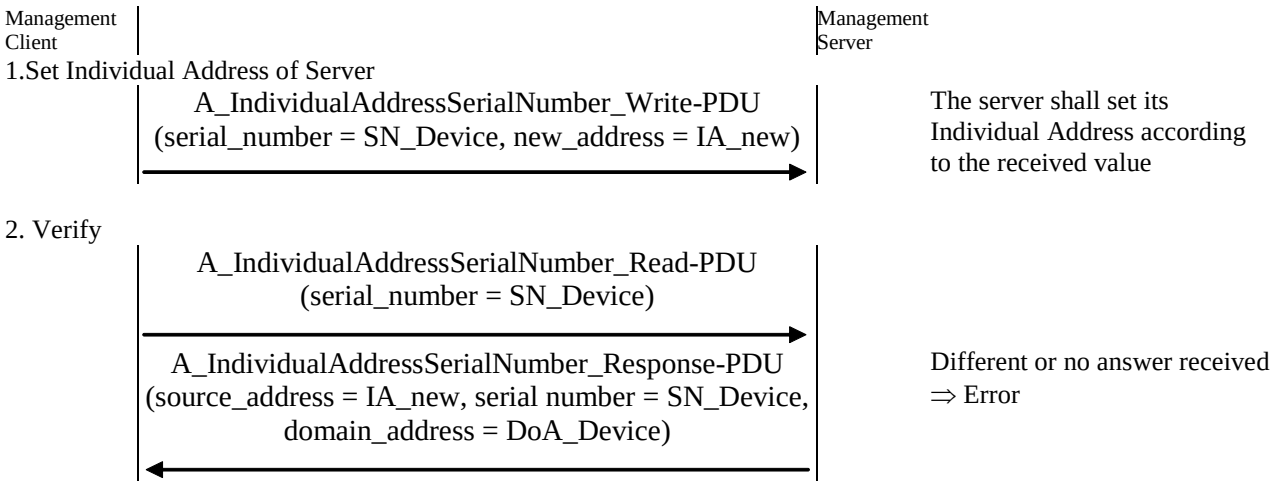
- A_IndividualAddressSerialNumber_Write
- A_IndividualAddressSerialNumber_Read

Parameters of the Management Procedure

NM_IndividualAddress_SerialNumber_Write(/* [in] */ SN_device, /* [in] */ IA_new, /* [out] */ DoA_Device)

SN_Device:	KNX Serial Number of device to which the Individual Address will be assigned.
IA_new:	Individual Address to be programmed.
DoA_Device:	The Domain Address of the device if the device is on an open medium supporting a Domain Address.

Sequence



NOTE - Opposite to the procedures NM_IndividualAddress_Write and

NM_DomainAndIndividualAddress_Write, both requiring that the Programming Mode be active in the device and using the service A_IndividualAddress_Write, this procedure **does not reset** the device after assigning the Individual Address.

Exception handling

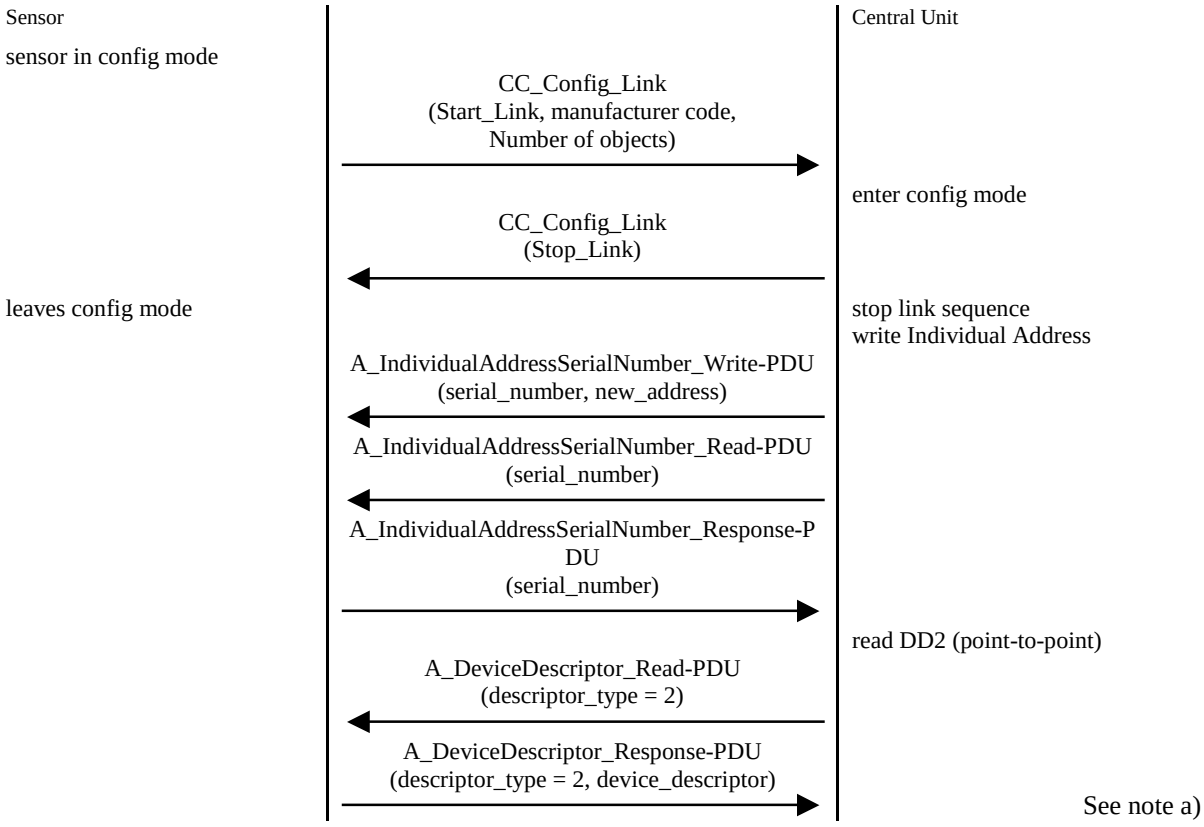
The default exception handling shall apply.

2.6 NM_IndividualAddress_SerialNumber_Write2

Use

NOTE The beginning of the procedure is identical to the link sequence of PB-Mode. A central Management Client shall react to the Start_Link command by sending a Stop_Link. The device shall stop the link sequence.

At this point the Management Client shall know the KNX Serial Number of the device and shall assign the Individual Address with A_IndividualAddressSerialNumber_Write. Finally the Device Descriptor shall be read.



Notes

a) In the context of this Management Procedure NM_IndividualAddress_SerialNumber_Write2, the A_DeviceDescriptor_Read-service is only applied to check whether the Sensor can be addressed using its new Individual Address. The Management Client is only interested in whether it receives a response or not; the contents, this is, the value of descriptor_type and device_descriptor should not be evaluated at this point.

2.7 NM_DomainAddress_Read

Use

This Network Management Procedure shall be used to read out the Domain Addresses of all the devices in which Programming Mode is active.

This procedure works independently of the configuration of the Domain Address and the Individual Address of the Router.

Used Application Layer Services for Management

- A_DomainAddress_Read

Parameters of the Management Procedure

NM_DomainAddress_Read (/* [out] */ individual_addresses[], /* [out] */ domain_addresses[])

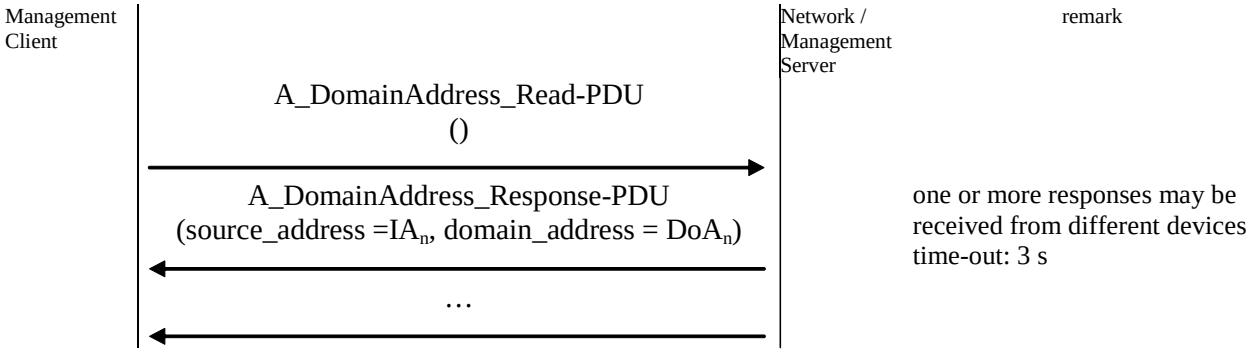
individual_addresses[]:	The collection of all the Individual Addresses of the devices in which Programming Mode is active.
domain_addresses[]:	The collection of all theDomain Addresses of the devices in which Programming Mode is active.

Variables

$IA_{n:}$ The IA of one device n that responds to the A_DomainAddress_Read-PDU. The Management Client shall collect all IA_n of the individual responses and report these via individual_addresses[].

DoA_n: The DoA_n of one device n that responds to the A_DomainAddress_Read-PDU. The Management Client shall collect all DoA_n of the individual responses and report these via domain_addresses[].

Sequence



Exception handling

The Management Client shall always wait until the time-out has elapsed. It shall collect all responses during this time-out.

- If no A_DomainAddress_Response is received, there is no device in which Programming Mode is active.
- If one A_DomainAddress_Response is received, there is exactly one device in which Programming Mode is active.
- If more than one response is received, there are several devices in which Programming Mode is active.
- If two or more responses with the same Domain Address and Individual Addresses are received, there is more than one device with the same Domain Address and the same Individual Addresses.

The Management Client shall not evaluate Layer-2 repetitions.

2.8 NM_DomainAndIndividualAddress_Read

Use

This Network Management Procedure shall be used to read the Domain Address and the Individual Address of one or more devices in which the Programming Mode is active.

Used Application Layer Services for Management

- A_DomainAddress_Read
- A_IndividualAddress_Read

Parameters of the Management Procedure

NM_DomainAndIndividualAddress_Read(/* [out] */ individual_addresses[],
/* [out] */ domain_addresses)

individual_addresses[]: The collection of all the Individual Addresses of the devices in which Programming Mode is active.

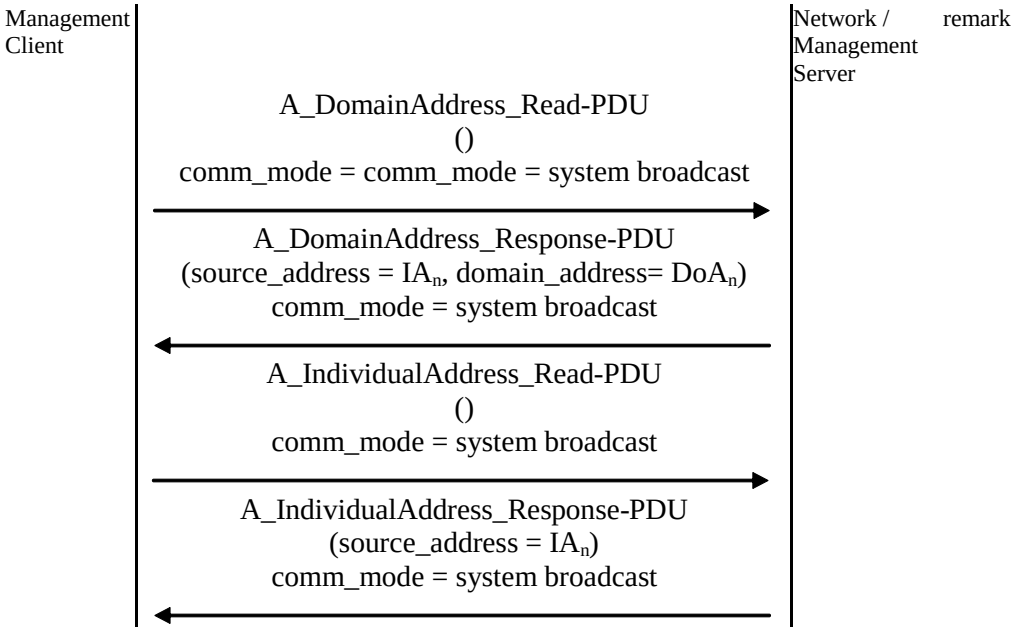
domain_addresses[]: The collection of all theDomain Addresses of the devices in which Programming Mode is active.

Variables

$IA_{n:}$ The IA of one device n that responds to the A_DomainAddress_Read-PDU. The Management Client shall collect all IA_n of the individual responses and report these via individual_addresses[].

DoA_n: The DoA_n of one device n that responds to the A_DomainAddress_Read-PDU. The Management Client shall collect all DoA_n of the individual responses and report these via domain_addresses[].

Sequence



2.9 NM_DomainAndIndividualAddress_Write

Use

This Network Management Procedure shall be used to set the Domain Address and the Individual Address of one single device that is in Programming Mode.

This procedure shall ensure that no other device has the same Individual Address and shall wait until there is exactly one device in which Programming Mode is active. It shall verify whether the programming is successful and shall switch deactivate the Programming Mode in the device into by executing a restart of the device.

For this procedure the Management Server has to provide a free Domain Address.

Used Application Layer Services for Management

- A_Connect
- A_DeviceDescriptor_Read
- A_DomainAddress_Read
- A_DomainAddress_Write
- A_IndividualAddress_Write
- A_Restart

Parameters of the Management Procedure

NM_DomainAndIndividualAddress_Write(/* [in] */ DoA_new, /* [in] */ IA_new)

DoA_new: The Domain Address to be assigned to the device.

IA_new: The new Individual Address to be assigned to the device.

Variables

IA_current: The current IA of the device in which Programming Mode is active prior to the assignment of IA_new.

DoA_current: The current IA of the device that in which Programming Mode is active prior to the assignment of IA_new.

Sequence

Management Client | Network / Management Server | remark

1. Verify whether the Individual Address IA_new is already occupied on the network.

A_Connect-PDU (destination_address = IA_new) →

if negative A_Connect.Lcon ⇒ IA_new is not occupied; continue with 2.
else (this is, a positive A_Connect.Lcon is received)

If the device that occupies the IA IA_new does not support Transport Layer connections, it shall send a T_Disconnect-PDU.

A_Disconnect-PDU (destination_address = IA_new) ⇠

If an A_Disconnect-PDU is received then IA_new shall be regarded as occupied; end procedure.
else (no A_Disconnect-PDU is received)

If a device that occupies IA_new is present on the network, and does support Transport Layer connections, it shall have no other reaction on the bus than the Layer-2 acknowledge that initiates the above A_Connect.Lcon

The A_DeviceDescriptor_Read-PDU shall use DD0.

A_DeviceDescriptor_Read-PDU (destination_address = IA_new, descriptor_type = 0000h) →

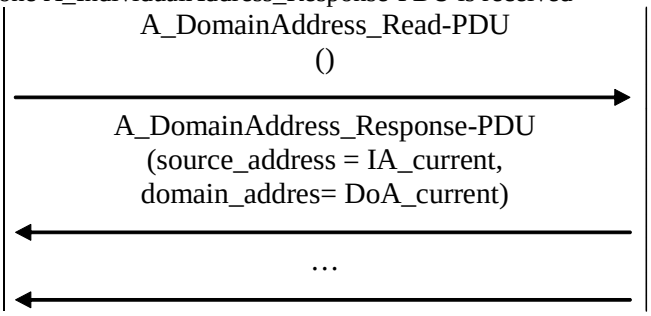
A_DeviceDescriptor_Response-PDU (descriptor_type, device_descriptor) ←

If the Management Client receives an A_DeviceDescriptor_Response-PDU it shall conclude that the Individual Address IA_new is *occupied*.

The Management Client shall accept any value of descriptor_type, also values ≠ 0.
If no A_DeviceDescriptor_Response-PDU is received after time-out
⇒ IA_new is not occupied
endif

A_Disconnect-PDU (destination_address = IA_new) →

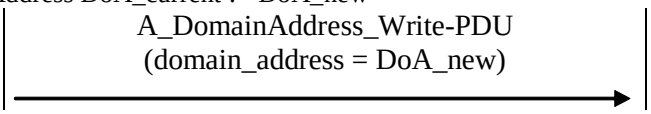
2. wait until Programming Mode is active in the device:
repeat until one A_IndividualAddress_Response-PDU is received



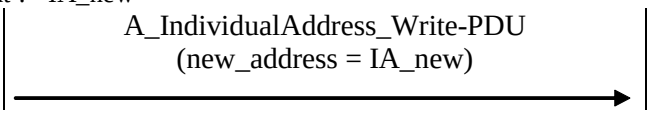
one or more responses may be received from different devices
time-out: 1 s

if more than one response is received ⇒ than Programming Mode is active in more than one device
end repeat

3.set Domain Address and Individual Address
if Domain Address DoA_current != DoA_new

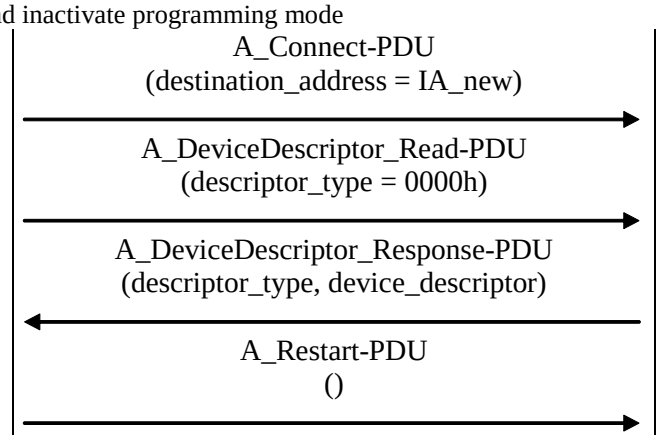


endif
if IA_current != IA_new



endif

4. verify and inactivate programming mode



Abort the connection of the Management Client side Transport Layer.

Exception handling

to 1.: If an A_Disconnect-PDU is received instead of an A_DeviceDescriptor_Response-PDU, then a device with this Individual Address exists but it may either already have another Transport Layer connection open and not accept any further Transport Layer connections, or does not support connection oriented communication mode.
⇒ The Management Client shall continue with the Management Procedure in every case.

to 2.: The Management Client shall always wait until the time-out has elapsed. It shall collect all responses during this time-out.
This Management Procedure shall wait until Programming Mode is active in only exactly one device [3].

The following cases may occur at this point.

- A device with the Individual Address exists, but it is not the one in which Programming Mode is active.
  ⇒ The Management Client shall not continue with the Management Procedure.
- A device with the Individual Address exists, and it is the one in which Programming Mode is active.
  ⇒ The Management Client shall continue with the Management Procedure.
- No device with the Individual Address exists.
  ⇒ The Management Client shall continue with the Management Procedure.

to 4.: If no A_DeviceDescriptor_Response-PDU is received, then the programming of the Individual Address may have failed or the system (Router) has not been configured correctly.

2.10 NM_DomainAndIndividualAddress_Write2

Use

This procedure NM_DomainAndIndividualAddress_Write2 differs from the above procedure NM_DomainAndIndividualAddress_Write in the following.

- It shall not check whether the Individual Address that shall be assigned is already present in the network. As a result, it does not need the service A_Connect.
- After assignment of the Individual Address it reads out the Device Descriptor connectionless.
- The Management Server is not restarted at the end of the procedure. As a result, it does not need the service A_Restart.

Used Application Layer Services for Management

- A_IndividualAddress_Read
- A_DomainAddress_Write
- A_IndividualAddress_Write
- A_DeviceDescriptor_Read

Parameters of the Management Procedure

NM_DomainAndIndividualAddress_Write2(/* [in] */ NmpDoANew, /* [in] */ NmpIANew,
/* [out] */ NmpIACurrent)

NmpDoANew:	The Domain Address to be assigned to the device
NmpIANew:	The new Individual Address to be assigned to the device.
NmpIACurrent:	Individual Address used by the device before the start of the Management Procedure.

[3] The user of the Management Client should get an information, how many devices are in Programming Mode (none or more than one)

Sequence

Management Client		Management Server (Device)	remark
			Enter Programming Mode (Manufacturer specific)

1. Wait until Programming Mode is active in the device.
   Repeat until one A_IndividualAddress_Response-PDU is received.

	A_IndividualAddress_Read-PDU () comm_mode = system broadcast →	
	← A_IndividualAddress_Response-PDU (source_address = NmpIACurrent)	

   The Management client shall store the KNX Serial Number of the device.
   If more than one response is received:
   ⇒ Programming Mode is active in more than one device
   end repeat

2. Set Domain Address

	A_DomainAddress_Write-PDU (domain_address = NmpDoANew) comm_mode = system broadcast →	
		The device shall store and use the received Domain Address NmpDoANew.

3. Set Individual Address
   if NmpIANew ≠ NmpIACurrent

	A_IndividualAddress_Write-PDU (new_address = NmpIANew) comm_mode = system broadcast →	
		The device shall store the received Individual Address.

   endif

4. verify

	A_DeviceDescriptor_Read-PDU (destination address = NmpIANew, descriptor_type = 0) comm_mode = point-to-point connectionless →	
	← A_DeviceDescriptor_Response-PDU (descriptor_type, device_descriptor) comm_mode = point-to-point connectionless	See note a)
	A_Restart-PDU () →	
		The device shall quit Programming Mode.

Notes

a) In the context of this Management Procedure NM_DomainAndIndividualAddress_Write2, the A_DeviceDescriptor_Read-service is only applied to check whether the Management Server (device) can be addressed using its new Individual Address. The Management Client is only interested in whether it receives a response or not; the contents, this is, the value of descriptor_type and device_descriptor should not be evaluated at this point.

2.11 NM_DomainAndIndividualAddress_Write3

This Network Management Procedure is not yet specified.

2.12 Procedures with A_DomainAddressSelective_Read

The Management Client (this is the local of Application Layer user) shall apply the A_Domain-AddressSelective_Read.req primitive to read the KNX PL110 - or KNX RF DoA of one or more devcies without using a specific Domain Address.

The type of network Resources that shall be read shall be indicated by field Type in the ASDU. The following network Resources can be read.

- Type 0: The Domain Address of one or more KNX PL110 MaS.
- Type 1: The Domain Address of one or more KNX RF MaS.

This service is particularly used to check the existence of any open media devices with the specified Domain Address in possibly neighbouring installations.

There is no common general behaviour specified for the MaS. The reaction of the MaS has to be specified case per case. In case the MaS receives an A_DomainAddressSelective_Read-PDU with a value of the field Type that it does not support, or with further service parameters for which no reaction is specified, then the MaS shall not react.

2.12.1.1 Type 00h – single octet DoA

The ASDU of the A_DomainAddressSelective_Read-PDU shall contain the following fields.

- type: This shall be the type of call of the A_DomainAddressSelective_Read-service. This field shall have the value 00h.
- start_address: This shall be the start_address of the range of Individual Addresses to which the Management Server shall compare its own Individual Address.
- range: This shall be the range of Individual Addresses, starting from start_address and ending at start_address + range to which a Management Server shall compare its own Individual Address.

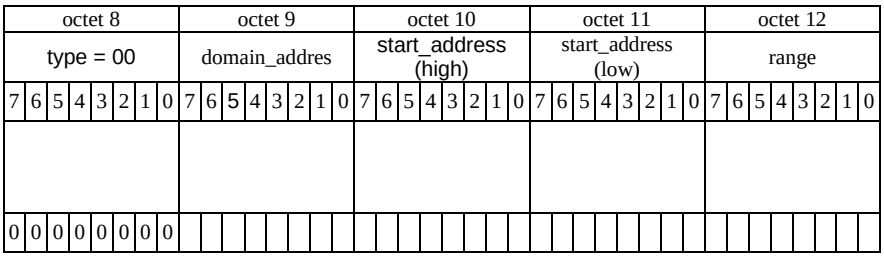


Figure 1 - A_DomainAddressSelective_Read-SDU – Type 00h (example)

The Management Server shall ignore the A_DomainAddressSelective_Read.ind primitive with *Type* 00h , if its Domain Address does not match with the argument domain_address, or its Individual Address is lower than the argument start_address or its Individual Address is higher than the (start_address + range).

If Management Server accepts the A_DomainAddressSelective_Read.ind primitive it shall respond to the Application Layer with an A_DomainAddress_Read.res primitive after a wait time: (individual_address - start_address) x T_{media} [4]. If the received argument range is lower than FFh and the Management Server receives during the waiting time an A_DomainAddress_Response-PDU then it shall terminate the transmission of its own response.

The A_DomainAddress_Response-PDU shall contain the fields Type and domain_address as show in Figure 2.

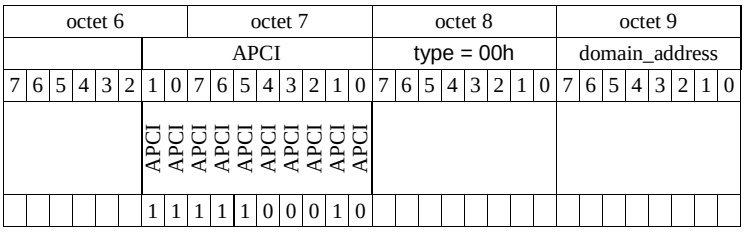


Figure 2 - A_DomainAddress_Response-PDU for a type 0 (1 octet DoA) (example)

2.12.1.2 Type 01h – six octet DoA

2.12.1.2.1 General requirements

The ASDU of the A_DomainAddressSelective_Read-PDU shall contain the following fields.

- type: This shall be the type of call of the A_DomainAddressSelective_Read-service. This field shall have the value 01h.
- domain_address_start: This shall be the start_address of the range of Domain Addresses to which the Management Server shall compare its own Domain Address.
- domain_address_end: This shall be the end_address of the range of Domain Addresses to which the Management Server shall compare its own Domain Address.

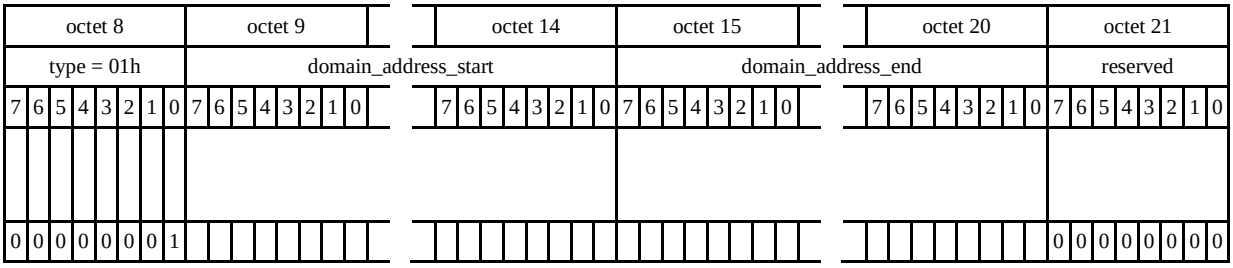


Figure 3 - A_DomainAddressSelective_Read-SDU – Type 01h (example)

The Management Server shall only accept the A_DomainAddressSelective_Read.ind primitive with *Type* 01h if its Domain Address is within the range domain_address_start to domain_address_end; else, it shall ignore the service.

If Management Server accepts the A_DomainAddressSelective_Read.ind primitive it shall respond to the Application Layer with an A_DomainAddress_Read.res primitive after a random wait time from 0 s to 2 s.

EXAMPLE 1 To have an equal spreading of the responses, this random wait time may for instance be based on the least significant octet to the KNX Serial Number of the Management Server device.

The A_DomainAddress_Response-PDU shall contain the fields Type and domain_address as show in Figure 4.

[4] T_{media} is specified in [05].

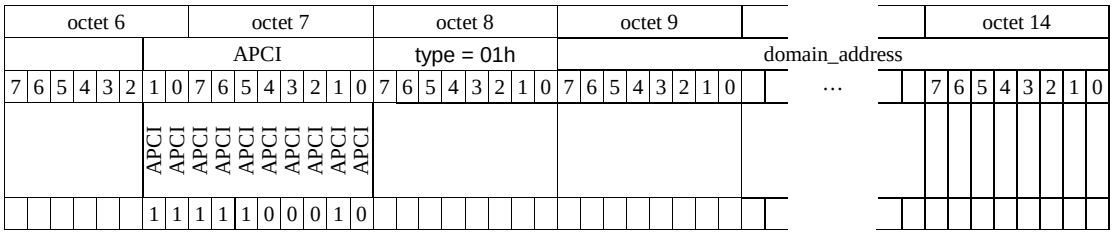


Figure 4 - A_DomainAddress_Response-PDU for a type 1 (6 octet DoA) (example)

2.12.1.2.2 NM_DomainAddress_Scan2

Use

This Management Procedure shall be used by a Management Client to scan for the presence of any devices with a 6 octet Domain Address that lies within a given range.

It returns whether or not there are any devices with a Domain Address in the scanned range. The devices respond with their Individual Address in the Source Address field of the responses, so the Individual Addresses are known by this as well. Additionally, on KNX RF, as the devices shall respond in system broadcast communication mode, the AET shall be 0 and the response shall contain the KNX Serial Number of the responding device. This is actually the key data retrieved by this procedure.

The MaC shall not execute this Management Procedure with values of any DoA in which the MSB differs from 00h.

Used Application Layer Services for management

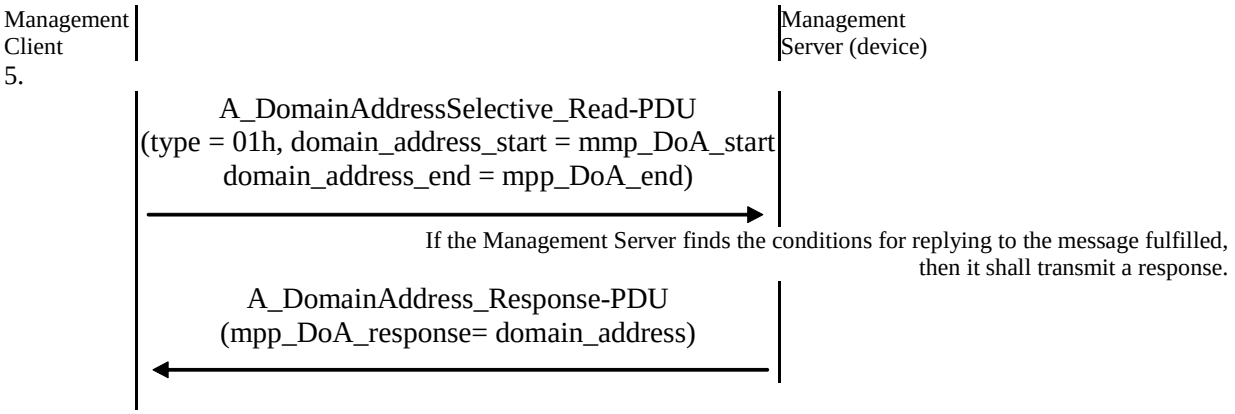
- A_DomainAddressSelective_Read
  (please note that this service responds with the A_DomainAddress_Response-PDU).

Parameters of the Management Procedure

NM_DomainAddress_Scan2(/* [in] */ mpp_DoA_start, /* [in] */ mpp_DoA_end,
/* [out] */ mmp_KNX_SN[],/* [out] */ mmp_IA[], /* [out] */ mpp_DoA_response[])

mpp_DoA_start:	This shall be lower limit of the range of Domain Addresses in which the presence of devices shall be searched.
mpp_DoA_end:	This shall be upper limit of the range of Domain Addresses in which the presence of devices shall be searched.
mpp_KNX_SN[]:	This shall be the collection of all KNX Serial Number values that have been used by the responding devices.
mpp_IA[]:	This shall be the collection of all Individual Address values that have been used by the responding devices.
mpp_DoA_response[]:	This shall be the DoA with which the Management Server has responded. There can be 0, 1 or multiple answers with the same of different DoA-values.

The A_DomainAddressSelective_Read-PDU shall be transmitted with priority *System*.



2.13 NM_Router_Scan

Use

This Network Management Procedure shall be used to determine what Routers are installed in a network.

The Management Client shall try to build up a connection to each possible Router. To this, it shall issue an A_Connect-PDU to each possible Router Individual Address. The Destination Address of this A_Connect-PDU shall be composed of:

- the Subnetwork Address field that shall start with 00h and be incremented by one for each next transmission of the A_Connect-PDU, and
- the Device Address field that shall have the fixed value 00h for each call of the A_Connect-PDU.

In this way, 255 A_Connect-PDUs will be transmitted.

The Management Client shall collect all A_Disconnect-PDUs. All Routers from which an A_Disconnect-PDU is received exist in the network.

For this procedure the Individual Address of the Routers and the Domain Address have to be configured.

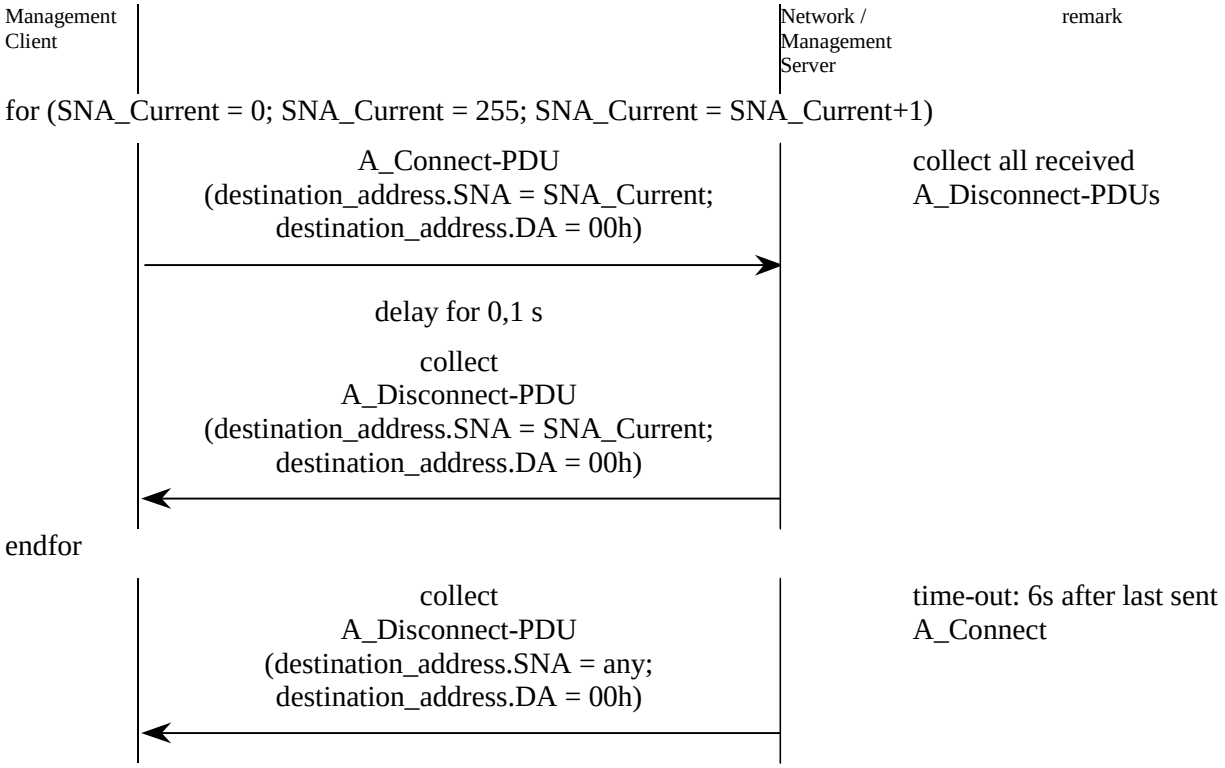
Used Application Layer Services for Management

- A_Connect

Variables

SNA_Current:	The current Subnetwork Address of the current Subnetwork in which the presence of a Router is searched.

Sequence



2.14 NM_SubnetworkDevices_Scan

Use

This Network Management Procedure shall be used to determine which devices exist on a Subnetwork.

The Management Client shall try to build up a connection using every possible correct Individual Address in this Subnetwork. It shall collect all A_Disconnect-PDUs. All devices from which an A_Disconnect-PDU is received shall be considered as existing on the Subnetwork.

For this procedure the Individual Address of the used Routers and the Domain Address have to be configured.

Used Application Layer Services for Management

- A_Connect

Parameters of the Management Procedure

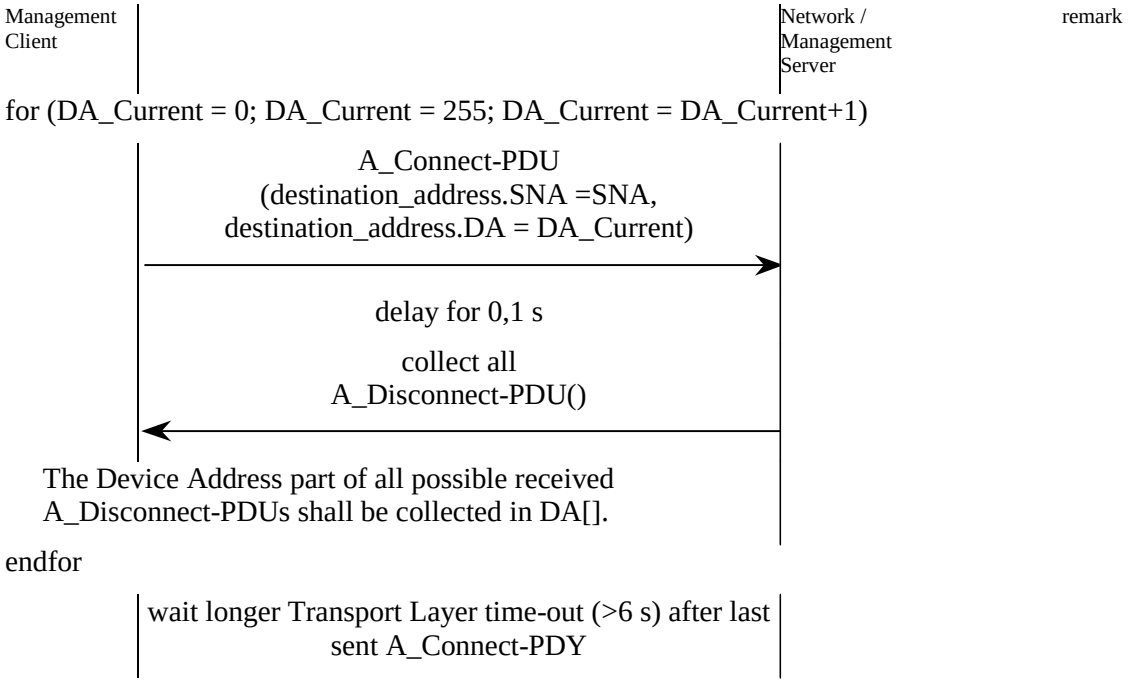
NM_SubnetworkDevices_Scan(/* [in] */ SNA, /* [out] */ DA[])

SNA:	Subnetwork Address of the Subnetwork in which the occupied Individual Addresses are to be scanned.
DA[]:	The collection of all Device Addresses of the devices discovered in the investigated Subnetwork.

Variables

DA_Current:	The current Device Address of which it will be checked whether a device with this Device Address exists on the Subnetwork that is being checked.

Sequence



2.15 NM_IndividualAddress_Reset

Use

This Network Management Procedure shall be used to reset the Individual Address of one or more devices in which Programming Mode is active to the default Individual Address FFFFh.

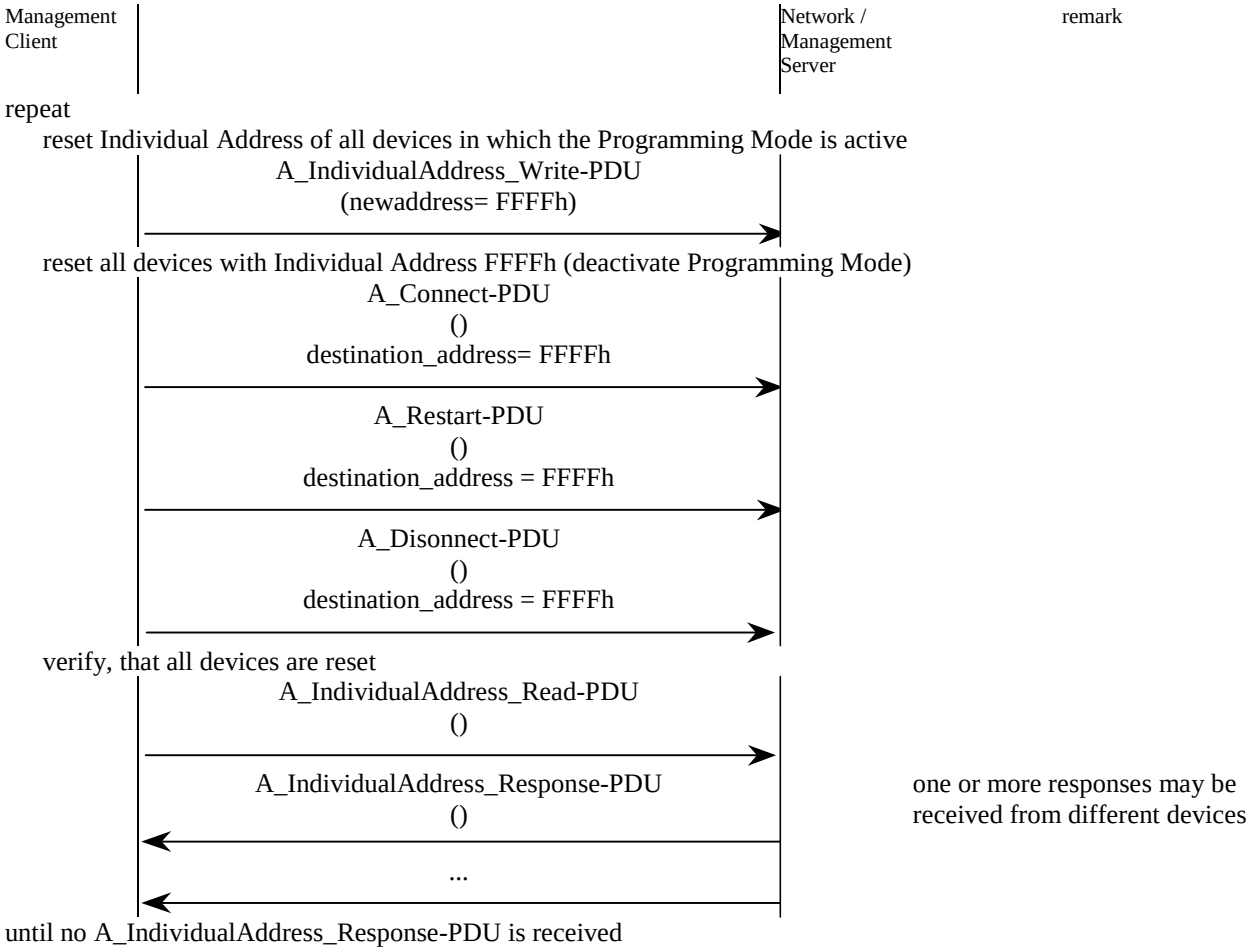
Used Application Layer Services for Management

- A_IndividualAddress_Write
- A_Restart
- A_IndividualAddress_Read
- A_Connect
- A_Disconnect

Parameters of the Management Procedure

This Management Procedure does not require any procedure parameters.

Sequence



Do not evaluate any local confirmation, or received telegrams, except the A_IndividualAddress_Read.Lcon and the A_IndividualAddress_Response-PDU.

2.16 NM_IndividualAddress_Check

NOTE This procedure has also been named NM_IndividualAddress_Scan.

Use

This Network Management Procedure shall be used by a network Management Client to check whether a given Individual Address is occupied on the network or not.

Used Application Layer Services for Management

- A_Connect
- A_DeviceDescriptor_Read
- A_Disconnect

Parameters of the Management Procedure

NM_IndividualAddress_Check(/* [in] */ IA_test, /* [out] */ result, /* [out] */ DDType, /* [out] */ DDx)

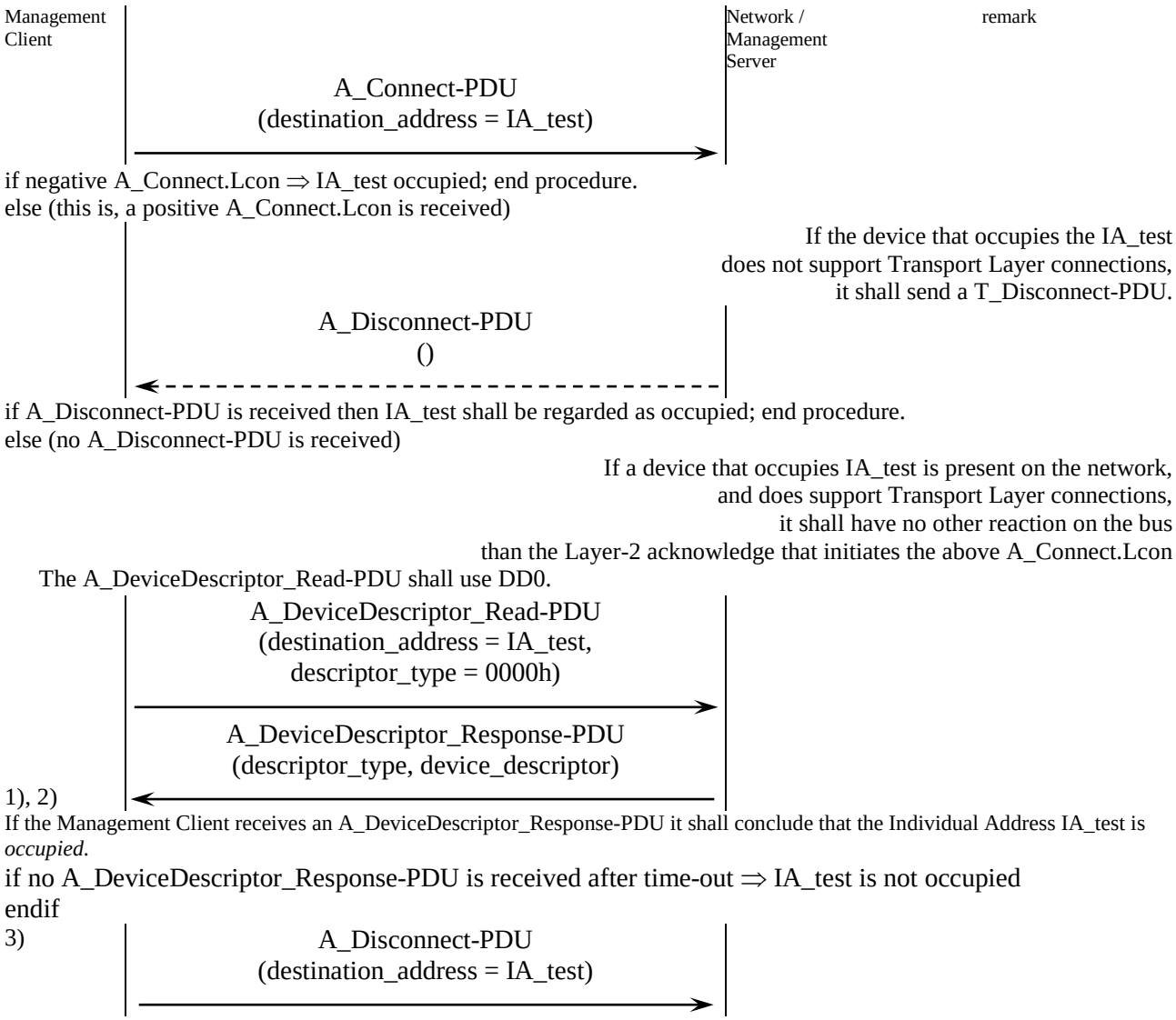
IA_test: Individual Address of which the occupation on the network has to be tested.

result: Result back to the user of the Management Procedure to indicate whether the IA_test is occupied on the network or not.

DDType: The Device Descriptor Type as reported by the device

DDx: The Device Descriptor value according the format DDType as reported by the device.

Sequence



Possible reactions

1) If the Network Management Server reacts on the connection oriented A_DeviceDescriptor_Read-PDU then the Management Client shall assume that the IA IA_test is occupied on the network and that the device that occupies this IA_test supports the connection-oriented Transport Layer.

2) The descriptor_type and the device_descriptor in the response by the device shall be reported back via DDType respectively DDx. The Device Descriptor Type may differ from 0 and the format of the Device Descriptor may be encoded accordingly as well.

3) If the Network Management Client receives an A_Disconnect-PDU and no A_DeviceDescriptor_-Response-PDU, then the Management Client shall assume that the IA_test is occupied on the network and that the device that occupies this IA_test does not support the connection-oriented Transport Layer.

2.17 Procedures with A_SystemNetworkParameter_Read

2.17.1.1 Introduction (informative)

This procedure shall be the alternative to the procedure NM_NetworkParameter_Read_R on system broadcast communication mode.

2.17.1.2 General Procedure

Precondition

This procedure shall be executed on system broadcast communication mode. This service is designed for the management (discovery, setting and diagnostics) of KNX open medium specific parameters. This is typically done at the beginning of the Configuration, when the Individual Addresses of the devices in the communication path between MaC and MaS have not yet been established. It is thus not possible to execute the procedure “Discovery of maximal Frame length” as specified in [04]. If using this procedure, the MaC shall thus make sure that the size request – and response-PDUs remains limited to an APDU of 14 octets at maximum.

Use

The MaC shall use this Management Procedure to find if a system- or device parameter of a given type and value is used in the network or not, using system broadcast communication mode on KNX open media.

Used Application Layer Services for Management

- A_SystemNetworkParameter_Read

The ASDU of the A_SystemNetworkParameter_Read-PDU shall contain the following fields.

- object_type: Value that shall be used by the MaC for the subfield object_type of the field parameter_type of the A_SystemNetworkParameter_Read-PDU.
- PID: Value that shall be used by the MaC for the subfield PID of the field parameter_type of the A_SystemNetworkParameter_Read-PDU.
- test_info: Value that shall be used by the MaC for the field test_info of the A_SystemNetworkParameter_Read-PDU.

octet 6								octet 7								octet 8								octet 9								octet 10								octet 11 ... n															
						APCI										parameter_type																								test_info															
																object_type																PID								reserved				operand							
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																																									
						0	1	1	1	0	0	1	0	0	0																											0	0	0	0								

Figure 5 - A_SystemNetworkParameter_Read-PDU (example)

The conditions for the MaS to respond to this service depend on the specific use and are given in the detailed procedures below.

- If the MaS receives an A_SystemNetworkParameter_Read PDU of which it does not support any of the service parameters (object_type, PID or further) or the reaction for these service parameters is not specified, then it shall not react.

- If the MaS finds the conditions for replying are not fulfilled, this is if the check of its investigated parameters against the test information is negative, then it shall ignore the service.
- If the MaS does accept the service, it shall respond with an A_SystemNetworkParameter_Read.res primitive after a random wait time. This random wait time is specified either per parameter_type in the detailed procedures below; additionally or alternatively, it is possible that the random wait time is communication by the MaC as part of the test_info. The data in the response shall depend on the network parameter type being read. The TSDU shall be an A_SystemNetworkParameter_Response-PDU as shown in Figure 6.
- The MaC shall not call this service with any service parameters (parameter_type, test_info) that are not specified in this paper.

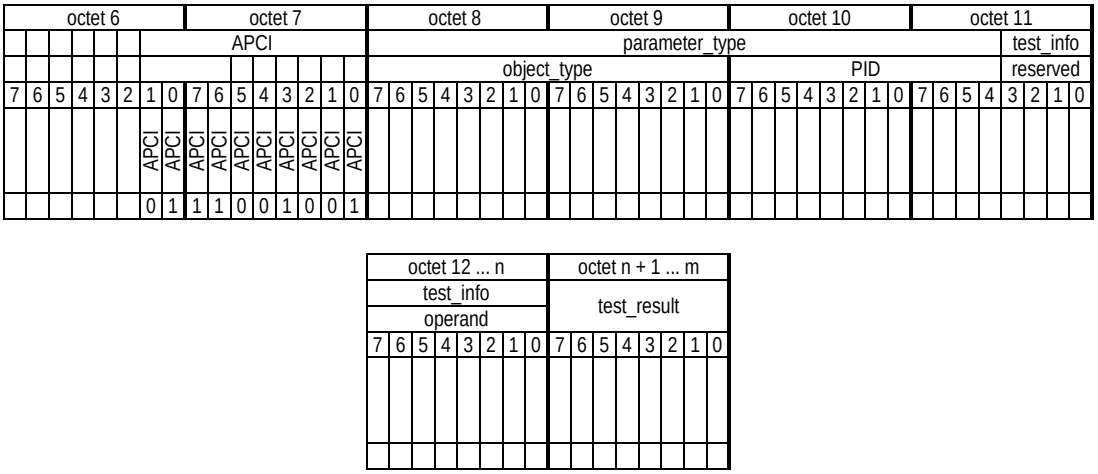


Figure 6 - A_SystemNetworkParameter_Response-PDU (example)

2.17.1.3 Overview of the accepted usage (informative)

All values are decimal, unless indicated otherwise. Indexes denote the bit length.

- **NM_Read_SerialNumber_By_ProgrammingMode**

  To read the KNX Serial Numbers of all devices in which Programming Mode is active.

  object_type: Device Object
  PID: 11 (PID_SERIAL_NUMBER)
  test_info: **operand:** **01h**
  The MaS shall respond only if its Programming Mode is active.
  test_result: KNX Serial Number of responder
  random wait time: constant: 0 s to 1 s

- **NM_Read_SerialNumber_By_ExFactoryState**

  To discover the devices in the network that have the factory default Domain Address and the factory default Individual Address.

  object_type: Device Object
  PID: 11 (PID_SERIAL_NUMBER)
  test_info: **operand:** **02h**
  The MaS shall respond only if its Domain Address and its Individual address both have their factory default value.
  wait_time: 1 octet: random wait time expressed in seconds (0 s to 255 s).
  Pending responses can be cancelled. Please check the specification of NM_Read_SerialNumber_By_-ExFactoryState.
  test_result: KNX Serial Number of responder
  random wait time: variable: contained in test_info

- **NM_Read_SerialNumber_By_PowerReset**

  To discover the devices in the network of that have just been powered off and on again.

  object_type: Device Object
  PID: 11 (PID_SERIAL_NUMBER)
  test_info: **operand:** **03h**
  The MaS shall respond only if it has just been powered on.
  wait_time: 1 octet: random wait time expressed in seconds (0 s to 255 s).
  Pending responses can be cancelled. Please check the specification of NM_Read_SerialNumber_By_-PowerReset.
  test_result: KNX Serial Number of responder
  random wait time: variable: contained in test_info

- **Manufacturer specific use of A_SystemNetworkParameter_Read**

  To perform manufacturer specific operations on system broadcast communication mode.

  object_type: any Interface Object Type: manufacturer and use specific
  PID: any Property Identifier: manufacturer and use specific
  test_info: **operand:** **FEh**
  The use and response conditions are implementation specific.
  manufacturer_id: Manufacturer Code of the MaS that may react to the service.
  The test_info may contain additional implementation specific fields.
  test_result: manufacturer and use specific
  random wait time: manufacturer and use specific
  The cancellation of pending responses is optionally possible; the methods are implementation specific without further requirements.

Other tests will be added in the future when needed.

2.17.1.4 Detailed procedure 1 – NM_Read_SerialNumber_By_ProgrammingMode

Use

This Network Management Procedure shall be used to read the KNX Serial Number of devices in which Programming Mode is active.

This procedure shall use system broadcast communication mode and is by that independent of the configuration of the Domain Addresses and the Individual Addresses of the devices and the (Media) Couplers.

The test_info shall consist of a single octet operand 01h.

octet 6								octet 7								octet 8								octet 9								octet 10								octet 11							
						APCI										parameter_type																								test_info			
																object_type																PID								reserved			
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	Device Object																PID_SERIAL_NUMBER																	
						0	1	1	1	0	0	1	0	0	0																									0	0	0	0				

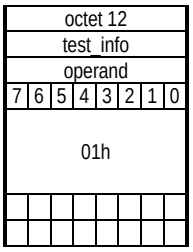


Figure 7 - A_SystemNetworkParameter_Read-PDU with NM_Read_SerialNumber_By_ProgrammingMode

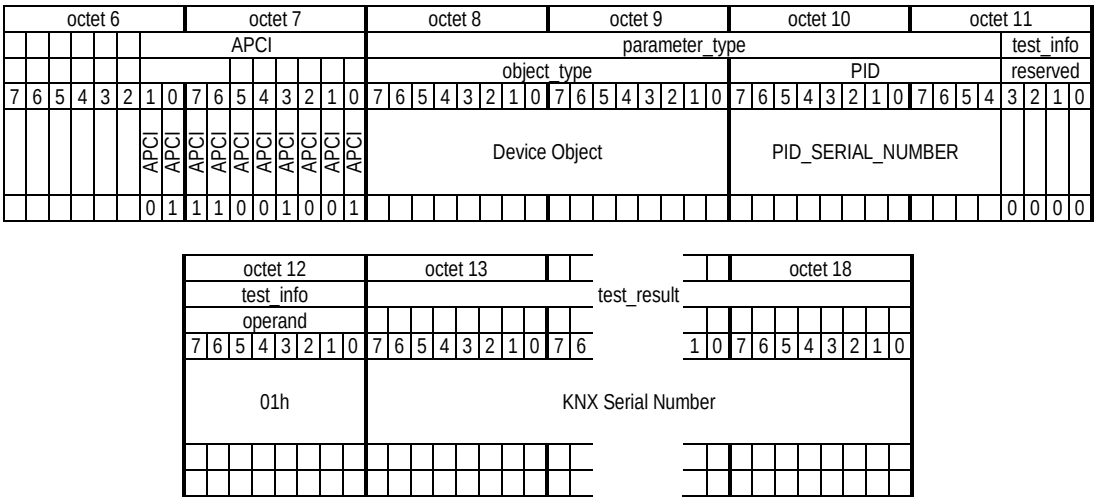


Figure 8 - A_SystemNetworkParameter_Response-PDU with NM_Read_SerialNumber_By_ProgrammingMode

Used Application Layer Services for Management

- A_SystemNetworkParameter_Read

Requirements to the MaS (device)

The MaS shall only reply to the A_SystemNetworkParameter_Read in this service, if its Programming Mode is active.

The random wait time for responding in this procedure shall be between 0 s and 1 s.

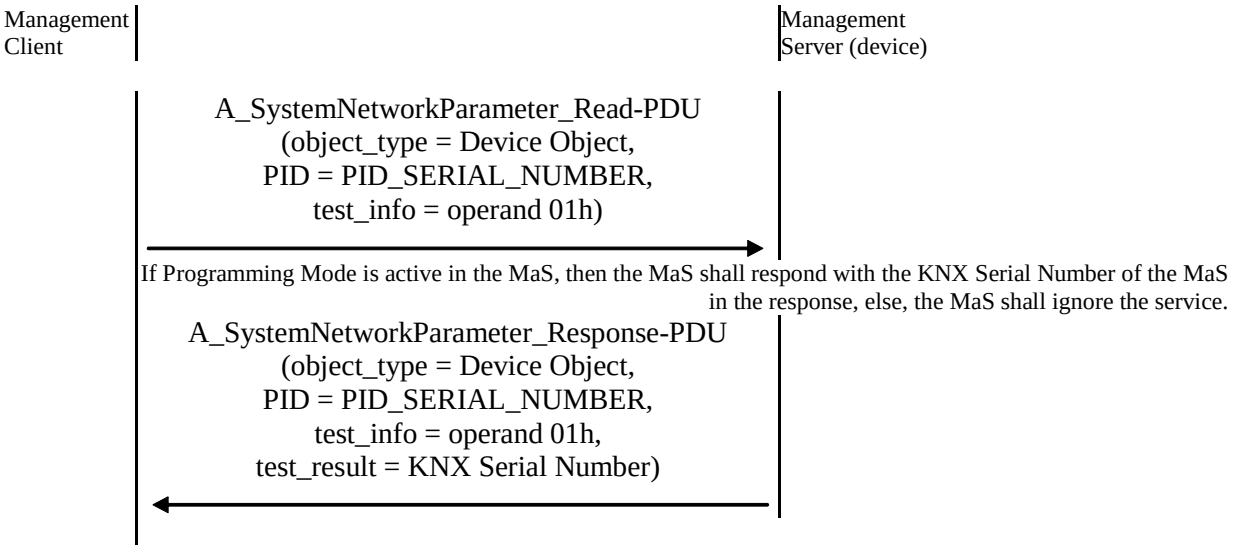
Parameters of the Management Procedure

NM_Read_SerialNumber_By_ProgrammingMode(/* [out] */ mpp_KNX_Serial_Number[])

mpp_KNX_Serial_Number[]	This shall be the list of KNX Serial Numbers of devices that respond to this procedure, that is, in which Programming Mode is active.

This Management Procedure does not have input parameters.

The A_SystemNetworkParameter_Read-PDU shall be transmitted with priority System.



NOTE 1 The value 01h of test_info identifies this specific use of A_SystemNetworkParameter_Read with the PID_SERIAL_-NUMBER. It shall not be mistaken for a comparison between the state of the Programming Mode ('1' = 'active') and the value 01h.

2.17.1.5 Detailed procedure 2 – NM_Read_SerialNumber_By_ExFactoryState

Use

This Network Management Procedure shall be used to scan for devices in the network of which both the Domain Address (if available) and the Individual Address have their factory default value.

The factory default values are specified here:

- for the DoA Realisation Type 1 (2 octets): in [11] clause 2.4.4.3.2.1.4 "Default value and Master Reset" in the specification of PID_PL110_DOA
- for the DoA Realisation Type 2 (6 octets): in [03] clause 3.2.4
- for the Individual Address: in [03] clause 3.3

This procedure shall use system broadcast communication mode and is by that independent of the configuration of the Domain Addresses and the Individual Addresses of the devices and the (Media) Couplers.

Used Application Layer Services for Management

- A_SystemNetworkParameter_Read

Requirements to the MaS (device)

The MaS shall only reply to the A_SystemNetworkParameter_Read-PDU in this service, if both its Domain Address (if available) and its Individual Address have the factory default value.

The random wait time for responding in this procedure shall be variable and shall be contained in the field wait_time in the A_SystemNetworkParameter_Read-PDU as specified in Figure 9.

If the MaS concludes on responding to this service request, then it shall delay its response until a random time in the period from 0 s to the number of seconds as indicated in the wait_time in the request.

- If during this delay the MaS receives a next A_SystemNetworkParameter_Read-PDU with the same parameters (object_type, PID and operand), but with the random wait time equal to 255 (FFh), then it shall cancel its response and not send an A_SystemNetworkParameter_Response PDU.
- If however the delay has already elapsed and the MaS has already requested the transmission of its request, then there are no further requirements.

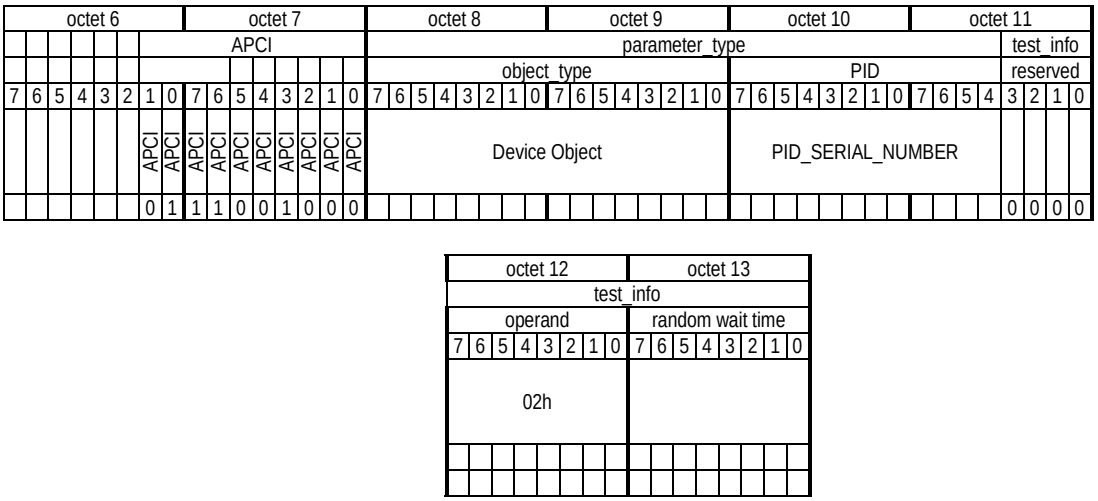


Figure 9 - A_SystemNetworkParameter_Read-PDU with NM_Read_SerialNumber_By_ExFactoryState

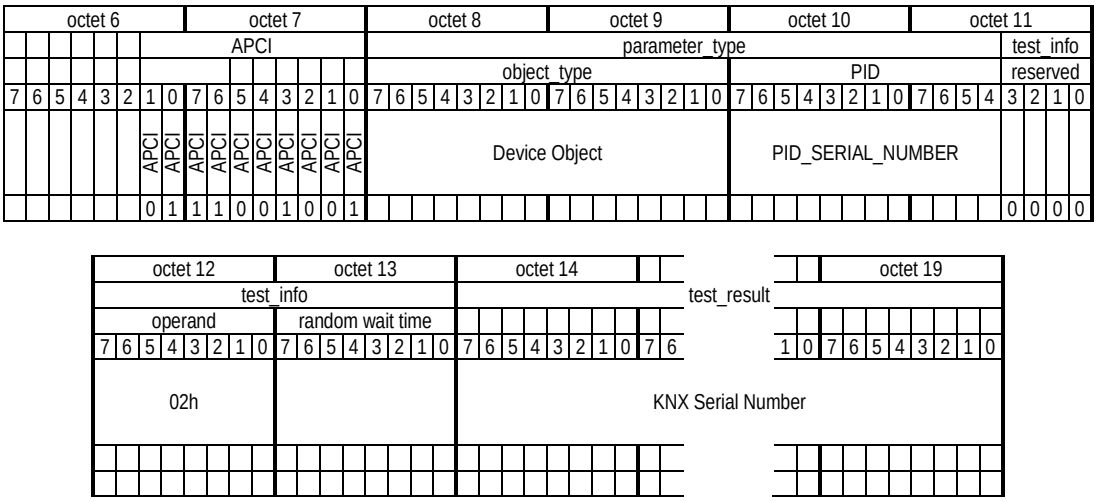


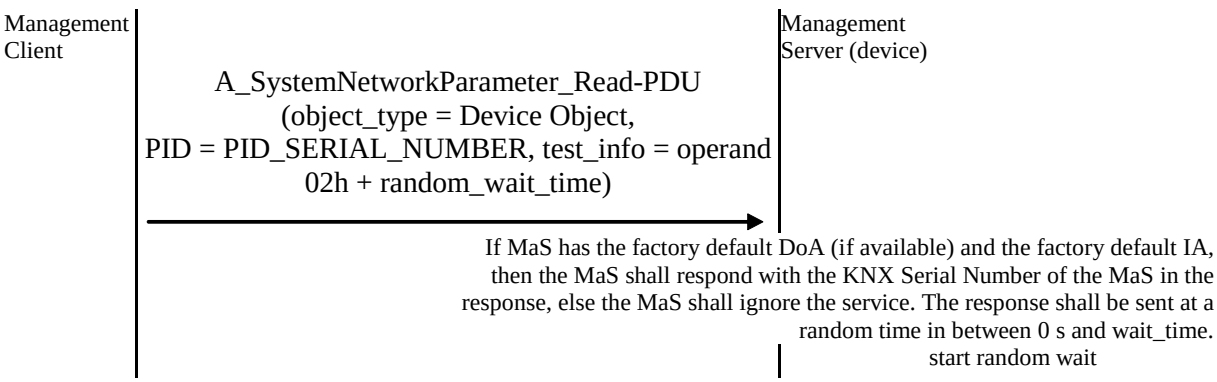
Figure 10 - A_SystemNetworkParameter_Response-PDU with NM_Read_SerialNumber_By_ExFactoryState

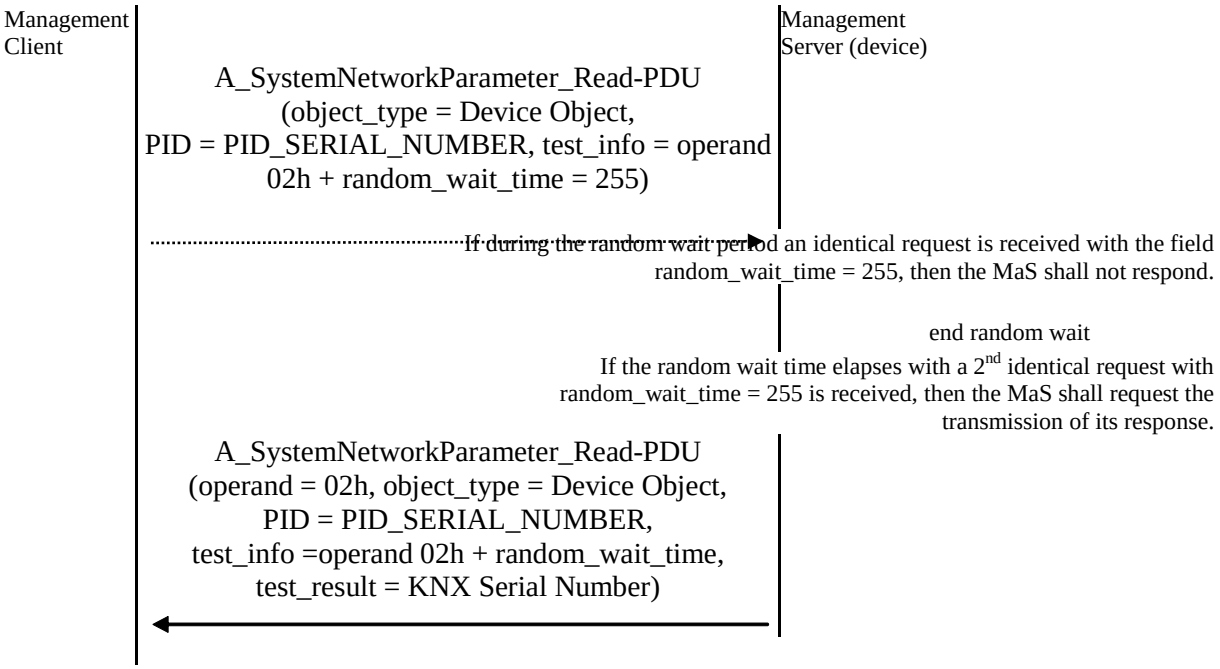
NM_Read_SerialNumber_By_ExFactoryState(/* [out] */ mpp_KNX_Serial_Number[])

mpp_KNX_Serial_Number[]	This shall be the list of KNX Serial Numbers of devices that respond to this procedure, that is, which have the factory default DoA and IA.

This Management Procedure does not have input parameters.

The A_SystemNetworkParameter_Read-PDU shall be transmitted with priority System.





Risks

- As this procedure uses system broadcast communication mode, if there are devices in ex-factory state outside the managed network, in neighbouring networks, these will respond as well.

2.17.1.6 Deatiled procedure 3 – NM_Read_SerialNumber_By_PowerReset

Use

This Network Management Procedure shall be used to scan for devices in the network of which have just been powered on.

This procedure shall help identifying and addressing “inaccessible devices”.

Opposite to the procedure NM_Read_SerialNumber_By_ExFactoryState, this procedure requires a human activity on the device and therefore has a better probability of excluding devices in neighbouring installations.

This procedure shall use system broadcast communication mode and is by that independent of the configuration of the Domain Addresses and the Individual Addresses of the devices and the (Media) Couplers.

Used Application Layer Services for Management

- A_DomainAddressSelective_Read

Requirements to the MaS (device)

The MaS shall only reply to the A_SystemNetworkParameter_Read-PDU in this service, if it has been powered on since less than 4 minutes. After 4 minutes, the device shall no longer react to this service with this operand. To these 4 minutes, there may be a tolerance of 30 seconds.

Additionally, the MaS shall not react if it has already replied to a preceding request since it has last powered on. This requires that the MaS keeps track of this.

If the MaS concludes on responding to this service request, then it shall delay its response until a random time in the period from 0 s to the number of seconds as indicated in the wait_time in the request.

- If during this delay the MaS receives a next A_SystemNetworkParameter_Read-PDU with the same parameters (object_type, PID and operand), but with the random wait time equal to 255 (FFh), then it shall cancel its response and not send an A_SystemNetworkParameter_Response PDU.
- If however the delay has already elapsed and the MaS has already requested the transmission of its request, then there are no further requirements.

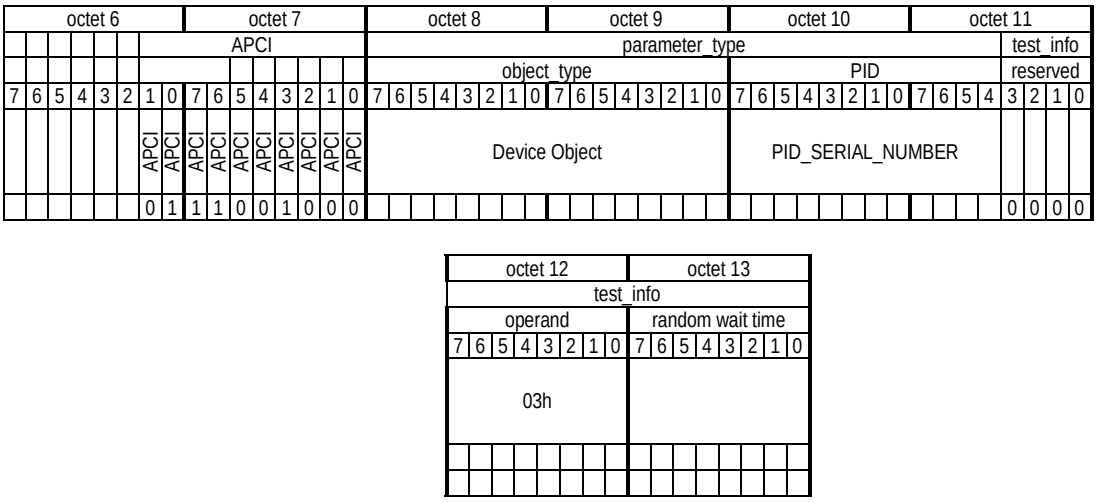


Figure 11 - A_SystemNetworkParameter_Read-PDU (Domain Parameter Read) with NM_Read_SerialNumber_By_PowerReset

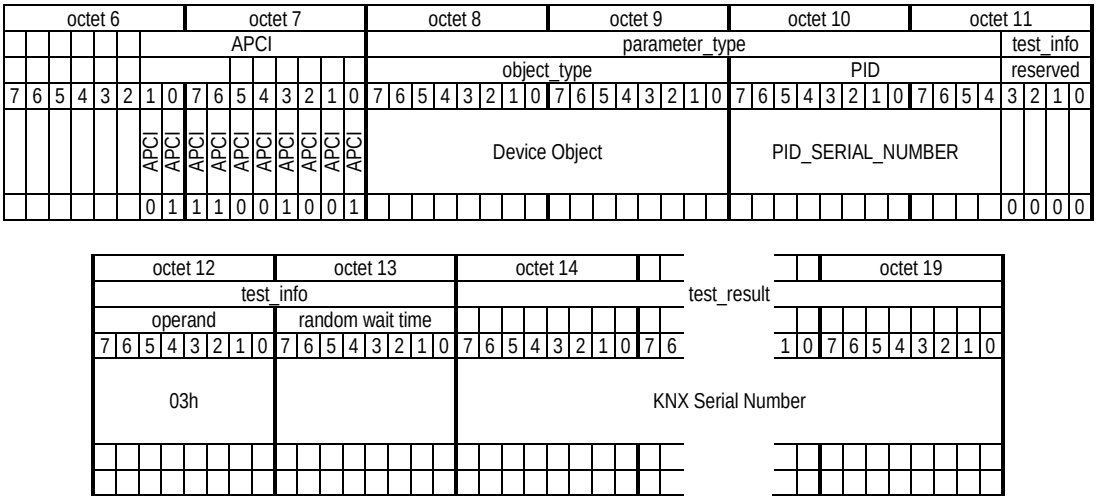


Figure 12 - A_SystemNetworkParameter_Response-PDU with NM_Read_SerialNumber_By_PowerReset

Requirements to the MaC

The MaC shall send the A_SystemNetworkParameter_Read-PDU in this procedure NM_Read_SerialNumber_By_PowerReset and repeat every 30 s during 4 minutes. It shall collect all the answers that arrive.

EXAMPLE 2 It is possible that the installer resets the MaS firstly and that the MaS completes its power up procedure before the installer triggers the procedure on the MaC (ETS). In this case, the MaS (device) will react immediately.

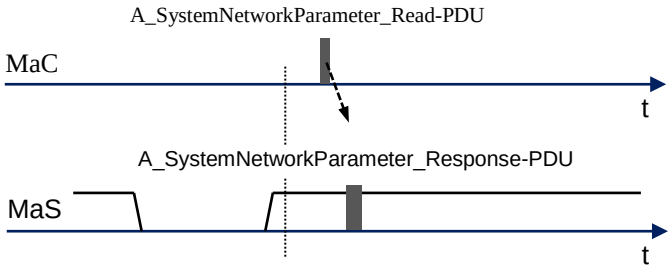


Figure 13 – The MaC triggers the procedure well after the MaS completes its power up

EXAMPLE 3 If the installer firstly triggers the procedure on the MaC (ETS) and then only resets the power of the MaS, then the MaS will not react.

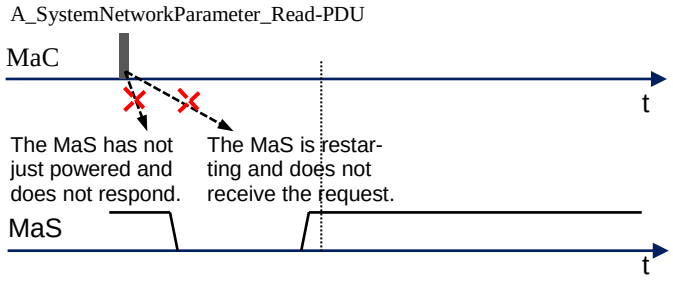


Figure 14 – The MaC triggers the procedure before the MaS completes its power up

Therefore, the MaC shall repeat the A_ SystemNetworkParameter_Read-PDU with a period of 30 s.

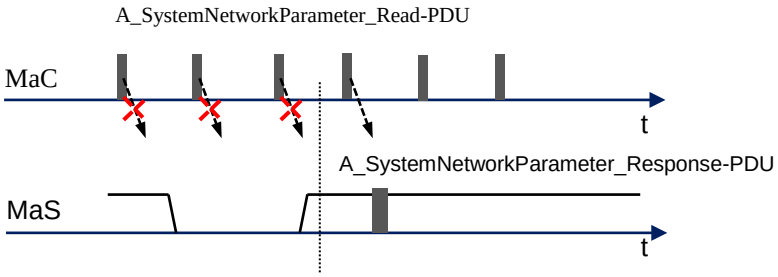


Figure 15 – The MaC repeats the request periodically. The MaS responds on the first request that it receives after power up.

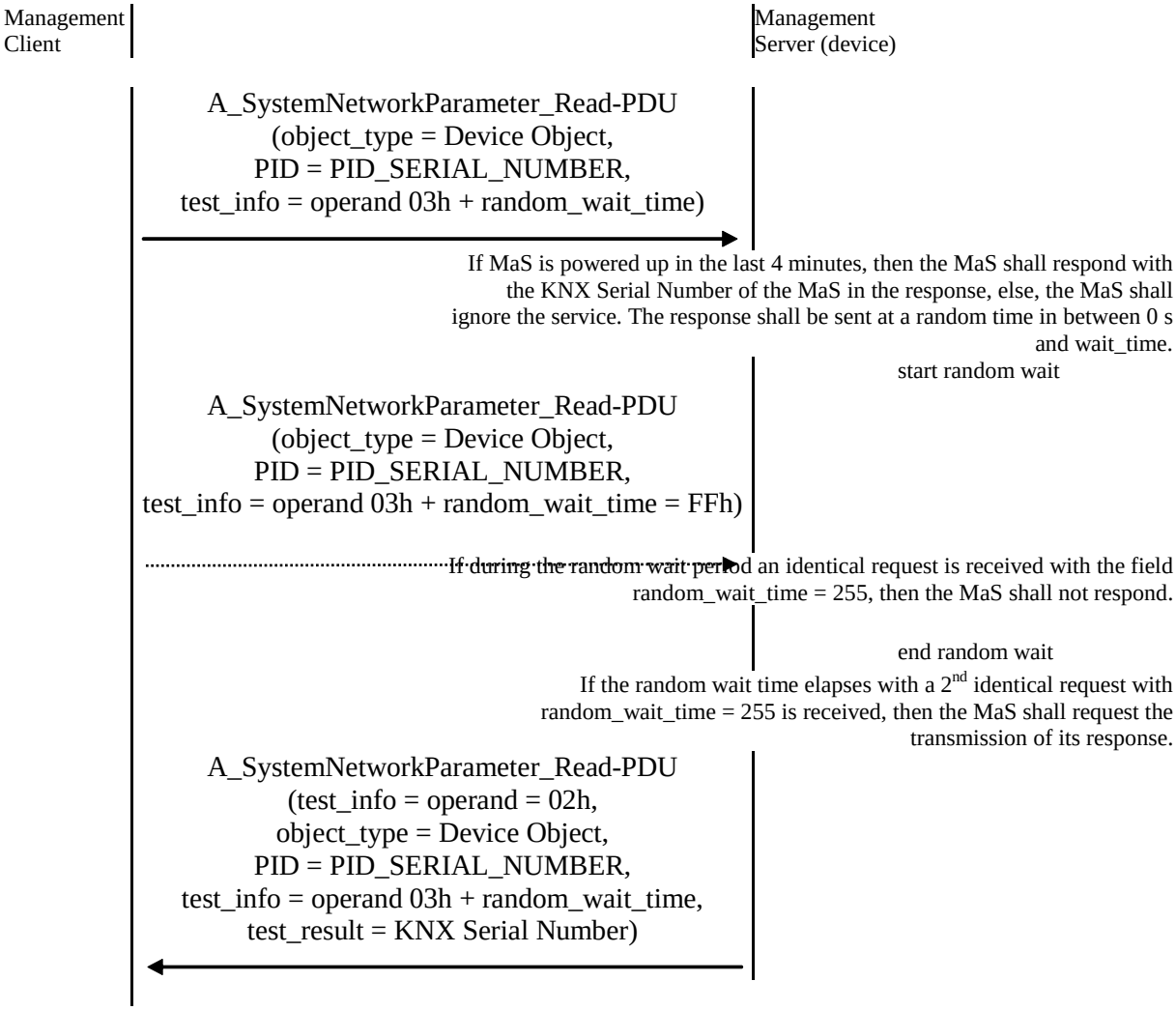
To make sure that the MaS reacts, the MaC repetition period shall be sufficiently smaller (30 s) than the after powering up during which the MaS reacts to the request.

NM_Read_SerialNumber_By_PowerReset(/* [out] */ mpp_KNX_Serial_Number[])

mpp_KNX_Serial_Number[]	This shall be the list of KNX Serial Numbers of devices that respond to this procedure, that is, which are powered up in the preceding 4 minutes.

This Management Procedure does not have input parameters.

The A_SystemNetworkParameter_Read-PDU shall be transmitted with priority System.



Risks

- As this procedure uses system broadcast communication mode, in the unlikely case that also devices outside the managed network, in neighbouring networks have just been powered on, these will respond as well.

2.17.1.7 Operand FEh – Manufacturer specific use of A_SystemNetworkParameter_Read

This procedure shall be used for manufacturer specific Network Configuration Procedures.

This allows for the manufacturer specific support of A_SystemNetworkParameter_Read

- for standard Properties for which no standard use of A_SystemNetworkParameter_Read is specified, and
- for non-standard Interface Objects and - Properties.

In the A_SystemNetworkParameter_Read-PDU, the ASDU shall be composed of the Interface Object Type, the PID, the operand FEh and the 2 octet manufacturer code; further manufacturer specific service parameters may follow.

The A_SystemNetworkParameter_Read-PDU for manufacturer specific use shall thus be formatted as specified in Figure 16.

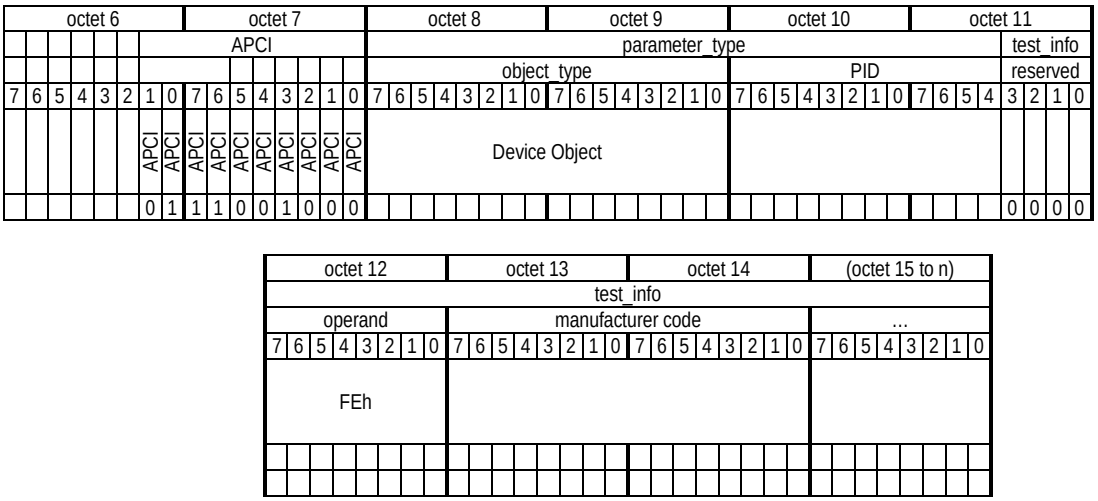


Figure 16 - A_SystemNetworkParameter_Read-PDU – Type FEh (example)

The A_SystemNetworkParameter_Response-PDU for this operand FEh shall repeat all the fields of the A_SystemNetworkParameter_Read-PDU which may be followed by further manufacturer specific response fields.

There are no further requirements concerning fields, use or timing for this manufacturer specific use. It is allowed to foresee the cancellation of pending responses by implementation specific means.

2.18 Procedures with A_SystemNetworkParameter_Write

2.18.1.1 General Procedure

Precondition

This procedure shall be executed on system broadcast communication mode. This service is mainly designed for the management (discovery, setting and diagnostics) of KNX open medium specific parameters. This is typically done at the beginning of the Configuration, when the Individual Addresses of the devices in the communication path between MaC and MaS have not yet been established. It is thus not possible to execute the procedure “Discovery of maximal Frame length” as specified in [04]. If using this procedure, the MaC shall thus make sure that the size request – and response-PDUs remains limited to an APDU of 14 octets at maximum.

Use

The MaC shall use this Management Procedure to set a system – or device parameter of a given type to a given value, using system broadcast communication model on KNX open media.

Used Application Layer Services for Management

- A_SystemNetworkParameter_Write

The ASDU of the A_SystemNetworkParameter_Write-PDU shall contain the following fields.

- object_type: Value that shall be used by the MaC for the subfield object_type of the field parameter_type of the A_SystemNetworkParameter_Write-PDU.
- PID: Value that shall be used by the MaC for the subfield PID of the field parameter_type of the A_SystemNetworkParameter_Write-PDU.
- value: Value that shall be used by the MaC and that shall be interpreted by the MaS when accessing the Property indicated in the object_type and PID.

2.18.1.2 Overview of the accepted usage (informative)

All values are decimal, unless indicated otherwise. Indexes denote the bit length.

- **Keep the bidirectional mode active in more than one KNX RF S-Mode device**

  To keep the bidirectional model enabled in KNX RF S-Mode semi-directional devices
  object_type: RF Medium Object
  PID: 60 (PID_RF_BIDIR_TIMEOUT)
  value: New value for the bidirectional mode time-out timer.

Please refer to the specification of PID_RF_BIDIR_TIMEOUT in [10]

2.19 Procedures with A_NetworkParameter_Write

2.19.1 NM_NetworkParameter_Write_R

Use

This Management Procedure shall be used by a Management Server to report the value of a parameter related to network configuration to any interested communication partner.

Used Application Layer Services for Management

- A_NetworkParameter_Write

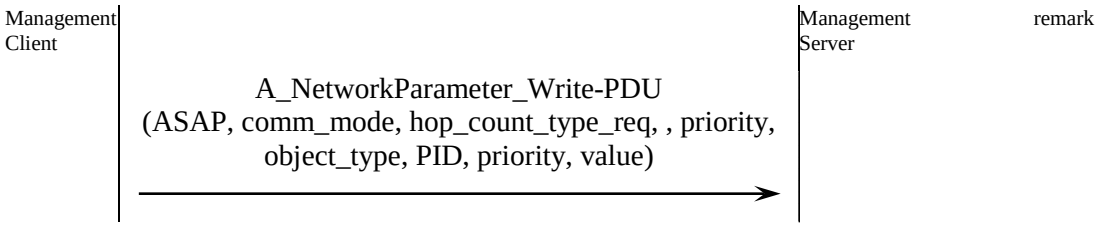
Parameters of the Management Procedure

NM_NetworkParameter_Write_R(/* [in] */ ASAP, /* [in] */ comm_mode, /* [in] */ hop_count_type_req, /* [in] */ object_type, /* [in] */ PID, /* [in] */ priority, /* [in] */ value)

ASAP:	The parameter ASAP shall only be evaluated if the parameter comm_mode equals point-to-point connectionless. It shall in this case contain the Individual Address of the communication partner to which the A_NetworkParameter_Write-PDU shall be sent.
comm_mode:	Communication mode that shall be used by the Management Server for transmission of the A_NetworkParameter_Write-PDU. It can be – point-to-all-points connectionless (this is broadcast), or – point-to-point connectionless.
hop_count_type_req:	Value of the hop_count that shall be used to transmit the A_NetworkParameter_Write-PDU. NOTE This value shall be fixed in function of the specific Property of which the value shall be reported via this Management Procedure as specified in [03].
object_type:	Value that shall be used by the Management Server for the subfield object_type of the field parameter_type of the A_NetworkParameter_Write-PDU.
PID:	Value that shall be used by the Management Server for the subfield PID of the field parameter_type of the A_NetworkParameter_Write-PDU.
priority:	The priority that shall be used for the transmission of the A_NetworkParameter_Write-PDU.
value:	This shall be the contents of the field value of the A_NetworkParameter_Write-PDU.

The A_NetworkParameter_Write-PDU shall be transmitted with priority *System*.

Sequence



2.19.2 NM_IndividualAddress_Check_LocalSubNetwork

Use

The procedure shall be used by a Management Client to check whether a given Individual Address is occupied on the Subnetwork where it is itself located (the “local” Subnetwork).

To check whether an Individual Address is occupied on a local Subnetwork the Management Client shall transmit an A_NetworkParameter_Write-PDU on point-to-point connectionless communication mode addressed at the Individual Address PPPP under test.

The result of this check will be *Occupied* or *NotOccupied*.

In order to have this procedure not perturb any other Subnetwork than the one on which the Management Client is mounted, the A_NetworkParameter_Write-PDU shall be sent with the parameter hop_count = 0, so that Routers do not pass this message.

The Subnetwork Address is assumed to be set.

Individual Address handling during the procedure

The Source Address in the A_NetworkParameter_Write-PDU used in the below sequence shall be set to the current Individual Address of the device, this is the one stored in NVRAM. During the procedure, the device shall acknowledge every telegram sent to this current Individual Address.

At that time, the Individual Address PPPP checked is not valid for the device. So the device shall not acknowledge any telegram sent to this Individual Address PPPP.

Possible situations and reactions during the Management Procedure

According to the specification of the Data Link Layer, the user layer of the MAC layer will receive an L_Data.con with the possible values of the parameter l_status:

- *ok*
- *not_ok.*

- Value *ok* means that a Layer-2 acknowledge has been received and that the Individual Address is occupied.
- Value *not_ok* means that:
  - a BUSY-acknowledge is received (on TP 1 only): there is a Management Server (device) that occupies the Individual Address that is checked but for internal reasons it cannot handle the frame transporting the A_NetworkParameter_Write-PDU.
  - a NACK-acknowledge is received (on TP 1 only): due to frame errors the frame transporting the A_NetworkParameter_Write-PDU has not been interpreted.
  - absence of acknowledge. This case demonstrates that the Individual Address is not occupied at this moment.

The probability that the flag l_status is set to *not_ok* , due to BUSY- or NACK-acknowledges, after the specified Layer-2 repetitions, can be neglected compared to the probability that this value *not_ok* is caused by the absence (no acknowledge) of a Layer-2 acknowledge. Therefore, the value *not_ok* shall always lead to the conclusion that the checked Individual Address is free. The cases where this is caused by unsuccessful transmission including BUSY and NACK retransmissions shall be recovered from by a constant Individual Address conflict detection procedure of the Configuration Mode that uses NM_IndividualAddress_Check_LocalSubnetwork.

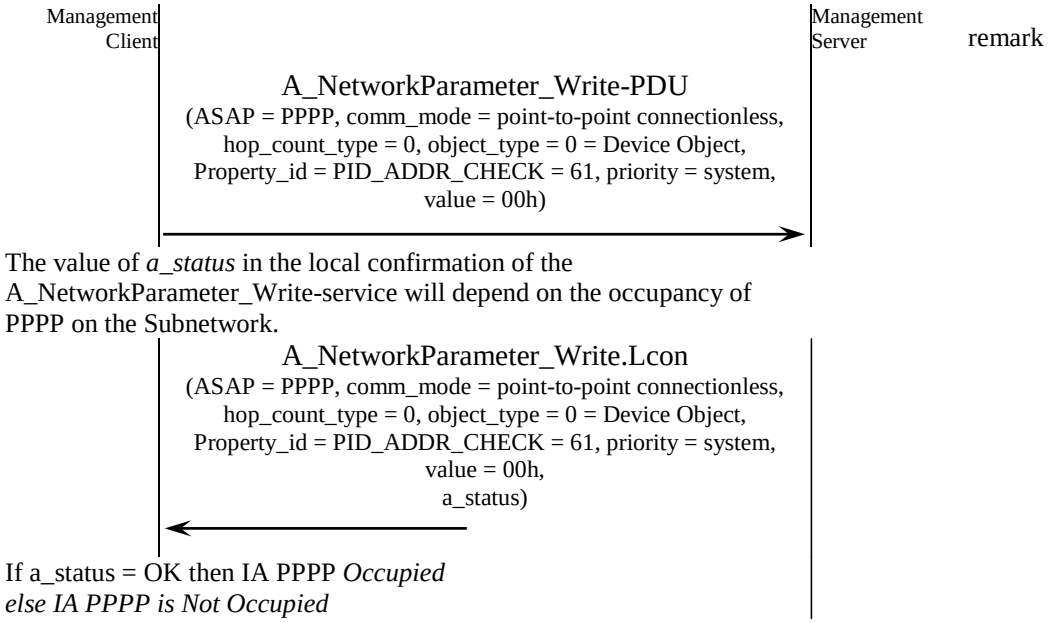
Used Application Layer Services for Management

- A_NetworkParameter_Write

Inputs

- PPPP: Individual Address of which the occupation on the Subnetwork has to be tested.

Sequence



2.19.3 NM_IndividualAddress_SerialNumber_Report

Use

This Management Procedure shall be used by a Management Server in order to announce its (new) Individual Address. The Management Server shall be identified by its KNX Serial Number.

The communication mode shall be broadcast. The hop_count_type shall be the default Network Layer value. The priority shall be set to “system”.

NOTE Most E-Mode devices that support this Management Procedure will have a KNX Serial Number and may report.

Used Application Layer messages for Management

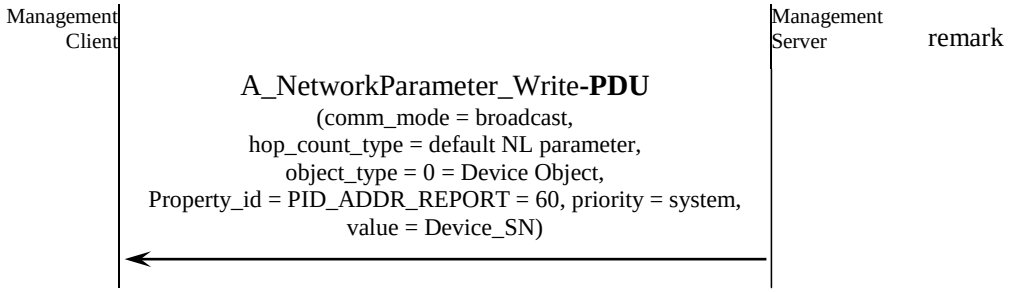
- A_NetworkParameter_Write

Parameters of the Management Procedure

NM_IndividualAddress_SerialNumber_Report(Device_SN)

Device_SN	KNX Serial Number of the device that reports its Individual Address

Sequence



2.20 Procedures with A_NetworkParameter_Read

2.20.1 General Procedure: NM_NetworkParameter_Read_R

Precondition

The use of the point-to-point connectionless communication mode for the response requires the preceding correct configuration of Individual Address in the communication path between requester and receiver(s). This shall be guaranteed by other preceding Management Procedures.

Use

This Management Procedure shall be used by a Management Client to find if a system- or device parameter of a given type and value is used in the network or not.

Used Application Layer Services for Management

- A_NetworkParameter_Read

Parameters of the Management Procedure

NM_NetworkParameter_Read_R(/* [in] */ ASAP, /* [in] */ hop_count_type_req, /* [in] */ object_type, /* [in] */ PID, /* [in] */ test_info, /* [in] */ comm_mode_req, /* [in] */ comm_mode_res, /* [in] */ hop_count_type_res, /* [out] */ result_data[])

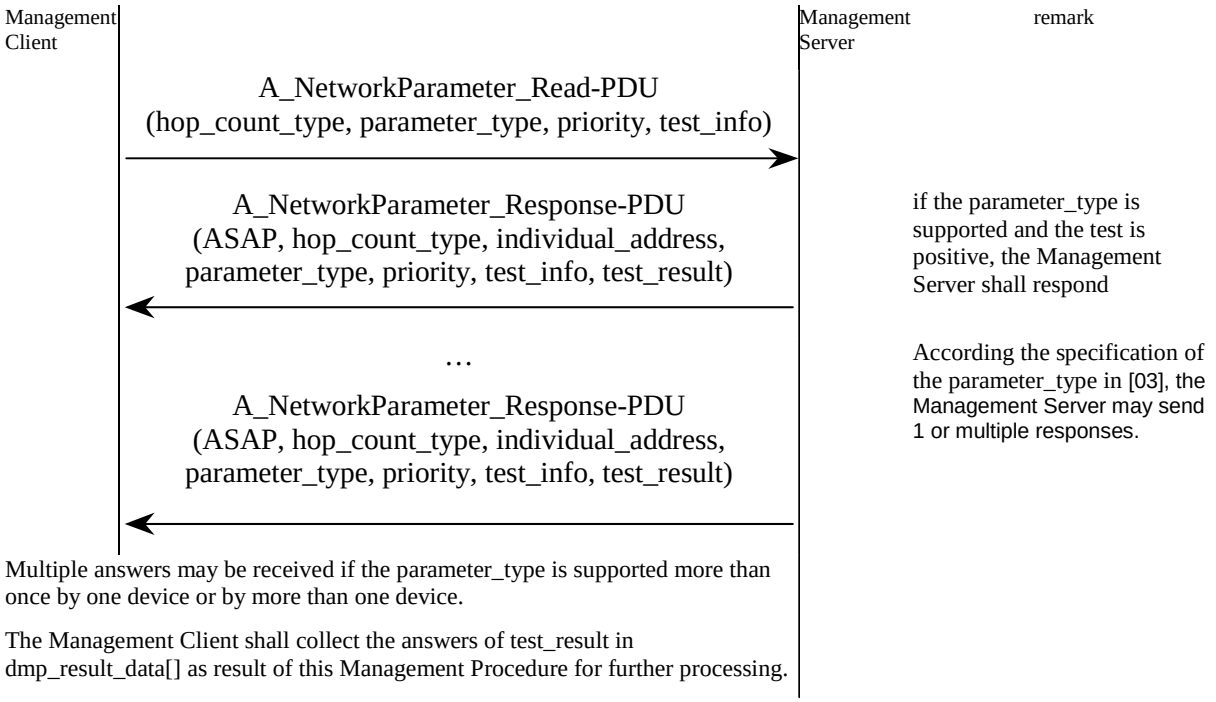
ASAP:	The parameter ASAP shall only be evaluated if the parameter comm_mode_req equals point-to-point connectionless. It shall in this case contain the Individual Address of the communication partner to which the A_NetworkParameter_Read-PDU shall be sent.
hop_count_type_req:	Value of the hop_count that shall be used by the Management Client for the transmission of the A_NetworkParameter_Read-PDU. NOTE This value shall be fixed in function of the specific Property that shall be accessible via this Management Procedure as specified in [03].
object_type:	Value that shall be used by the Management Client for the subfield object_type of the field parameter_type of the A_NetworkParameter_Read-PDU.
PID:	Value that shall be used by the Management Client for the subfield PID of the field parameter_type of the A_NetworkParameter_Read-PDU.

test_info:	Value that shall be used by the Management Client for the field test_info of the A_NetworkParameter_Read-PDU.
comm_mode_req:	Communication mode that shall be used by the Management Client for transmission of the A_NetworkParameter_Read-PDU. It can be – point-to-all-points connectionless (this is broadcast), or – point-to-point connectionless.
comm_mode_res:	Communication mode that shall be used by the Management Server(s) for transmission of the A_NetworkParameter_Response-PDU.
hop_count_type_res:	Value of the hop_count that shall be used by the Management Server(s) for the transmission of the A_NetworkParameter_-Response-PDU. NOTE This value shall be fixed in function of the specific Property that shall be accessible via this Management Procedure as specified in [03].
result_data[]:	The collection of all test_result data collected by the Management Client as result of the Management Procedure

Service parameters

test_info:	Value that shall be used by the Management Server(s) for the field test_info of the A_NetworkParameter_Read-PDU. NOTE This value shall be fixed in function of the specific Property that shall be accessible via this Management Procedure as specified in [03].

Sequence



Multiple answers may be received if the parameter_type is supported more than once by one device or by more than one device.

The Management Client shall collect the answers of test_result in dmp_result_data[] as result of this Management Procedure for further processing.

Management Server support

The remote Application Layer User, this is the management in the remote device, may respond with either no, one or several responses.

0. **No** response:
   If the parameter_type is not supported or the test_info leads to a negative result, the device shall not answer.

1. **One** response:
   If the parameter_type can, due to its nature, only be supported once by the device. This is the case for instance for a group address. This Link Layer parameter can be shared inside the device between multiple Group Objects, but is supported only once.
2. **Several** responses:
   If the parameter_type can be supported several times in the device. This can be the case for the Functional Block scan: a device can have the identical Functional Block supported more than once. E.g. a double-channel heating regulator, a double audio cassette-deck.

Error handling

Wait until the time-out has expired. This time-out is the sum of the following and has to be taken into account for each KNX device (Management Client, Management Server, Couplers…) and for each physical segment in the communication path between Management Client and Management Server.

- The maximum delay time in the device.

  This is the time that the Management Server in the device needs to handle this request. This may depend on the specific Network Parameter that is read.

- Random wait times in the device.

  In order to avoid a very high busload due to the usage of this procedure, in many cases it may be specified that the Management Server in the device delays its response by a random wait time, in order to spread in time the responses by many devices.

- Bus access times.

  This is the time that may be needed to access the bus, due to bus free detection and possible collision avoidance if other devices are transmitting.

- Medium times

  This is the actual time that the telegram occupies the medium and the short signalling difference between sender and receiver.

- Delay times in Couplers.

  This is the time that Repeaters, Line- and Backbone Couplers, Media Couplers and Interfaces may need to receive, evaluate and forward a telegram.

Therefore, the time-out should be defined for each specific use case of this Management Procedure.

If no response is received, the Management Client shall assume the system or device parameter for the checked value to be free.

2.20.2 NM_GroupAddress_Scan

Use

This Network Management Procedure shall allow testing whether a Group Address is used within the range start_address to start_address+range-1. To check for a single Group Address, the range shall be set to 1. The service request shall be handled via broadcast communication mode.

This Network Management Procedure shall be used by a Management Client to find if a Group Address is used in the network or not. In general, it shall be used to find a free Group Address before assigning it in a subsequent Management Procedure to one or more Management Servers (devices).

Used Application Layer Services for Management

- A_NetworkParameter_Read

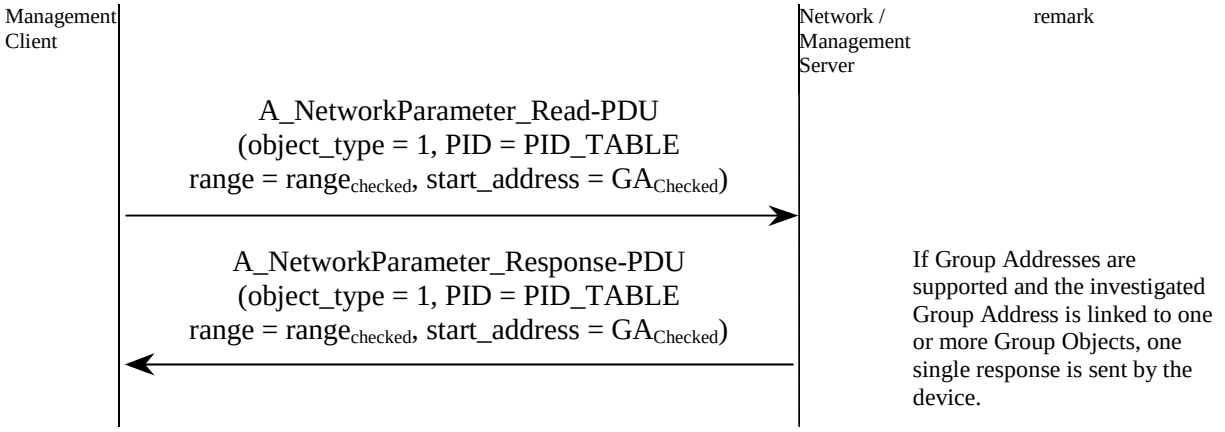
Parameters of the Management Procedure

NM_GroupAddress_Scan (/* [in] */ $GA_{checked}$, /* [in] */ $ranged_{chekced}$, /* [out] */ test_result)

Service parameters

object_type:	1 (Group Address Table)
PID:	PID_TABLE = 23 (List of Group Addresses)
comm_mode response:	point-to-point, connectionless

Sequence



Management Server support

If a Management Server (device) supports Group Addresses (multicast communication mode) and supports this service, it shall support this service without any limitation. When thus receiving this request, the Management Server shall test the contained Group Address(es) against all the Group Addresses it was assigned to, independent of their value, usage or status.

- Number of Responses

If the number of Group Addresses supported by the Management Server, within the range $GA_{Checked}$ to $GA_{Checked}+range_{checked}-1$

= 0 then the Management Server device shall send no answer

≠ 0 then the device shall send 1 single response
even
- if one Group Address is assigned to more than one Group Object, or
- if more than one Group Address within the indicated range is supported, or
- if the Group Address appears in the Group Address Table of the receiver but no associations to it exist in the Group Association Table.

- Response

There is no test-result field in the response PDU. The response shall be sent using point-to-point connectionless communication mode.

NOTE The field test_info (= range and Group Address) from the request shall be repeated in the response.

Management Client support

- Error Handling

The Management Client shall wait until the time-out has expired. This time-out is the sum of the maximum delay time in the device, in function of the medium and the frame transmission times. If no response is received, the Management Client shall assume that no Group Address in the range $GA_{checked}$ to $GA_{checked}$ + $range_{checked}$-1 is used in the network. If one or more responses are received, the Management Client shall conclude that at least one Group Address in the range the $GA_{checked}$ $GA_{checked}$ + $range_{checked}$-1 is be occupied.

- Usability of this Management Procedure

The support of the A_NetworkParameter_Read service is not mandatory for all Profiles. This means that if a Group Address is checked in a range that is typically used by devices of a Profile that does not require the support of this service, the result of this procedure may be less reliable.

2.20.3 NM_ObjectIndex_Read

This Network Management Procedure shall allow discovery of the index of a certain Interface Object Type in one device. The service request shall be handled via point-to-point communication mode.

This Network Management Procedure shall be used by a Management Client to find the Object Index(es) of one Object Type in one device.

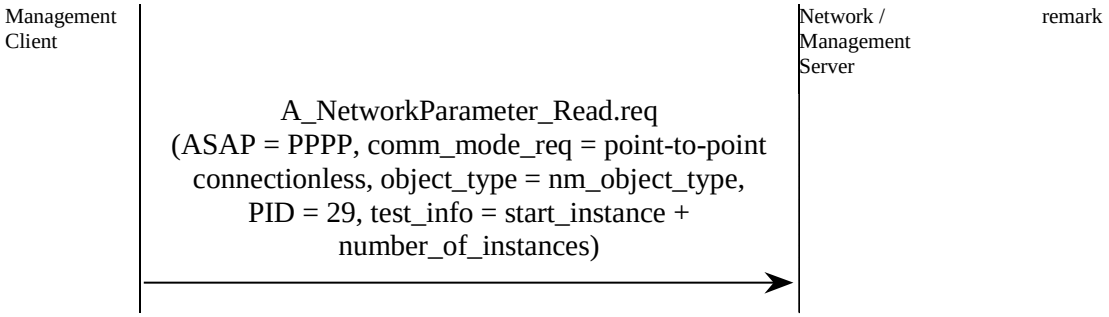
NM_ObjectIndex_Read (/* [in] */ ASAP, /* [in] */ comm_mode_req, /* [in] */ object_type, /* [in] */ PID, /* [in] */ test_info, /* [out] */ test_result, /* [in] */ comm_mode_res)

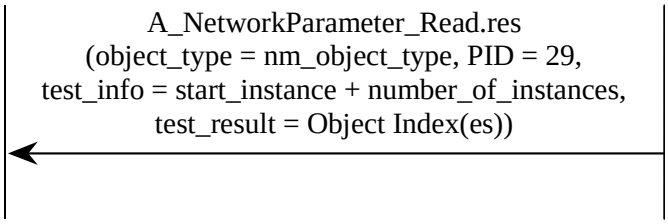
ASAP	PPPP = Individual Address of the Device in which the index of an Interface Object Type shall be discovered
comm_mode_req	point-to-point, connectionless
object_type:	nm_object_type: Interface Object Type of which the presence in the Management Server is to be discovered
PID:	PID_OBJECT_INDEX = 29 (see [03])
test_info:	Octet 11: start_instance of object_type
	Octet 12: number_of_instances
test_result:	Octet 13 … N: Object Index(es) of target object_type
comm_mode_res:	point-to-point, connectionless

Used Application Layer services for Management

- A_NetworkParameter_Read

Sequence





Only one response shall be given to one read request. If one response is not enough to deliver all indexes, than the Management Client shall read again with a higher start instance within the test_info.

2.20.3.1 Management Server support

If a Management Server receives this request, it shall answer with the total number of instances of object_type that are present in the Management Server device. Further, it shall answer with the requested number of Object Indexes.

The indexes in the response shall be sorted in ascending order. The numbering of instances shall with ‘1’, i.e. an Object Instance ‘0’ shall never exist. Instances of the same Object Type shall be numbered in ascending order of their indexes, i.e. instance 1 shall be the Interface Object with the lowest index of object_type, instance 2 shall be the interface with the next higher Object Index of the same object_type etc.

If a Management Server receives this request that is in point-to-point communication mode it shall answer with only one response. If not all requested indexes can be delivered in one response due to e.g. max. frame length, then the Management Server shall deliver as many indexes as fit in the frame with number_of_instances set to the number of instances delivered with the response. The Management Client can detect the total number of instances by setting start_instance and number of instances each to ‘0’ in the request, refer to case (d) in the table below (2.20.3.3).

2.20.3.2 Management Client support

If a Management Client does not know whether the Management Server supports long frames or not, it shall send a request only for 9 or less Object Instances at once. The response shall then fit in a short frame.

If a Management Client wants to know the indexes of all Object Instances and not all of them fit in one frame, then it shall repeat the procedure. It shall then send the repeated request with an adjusted test_info (higher value for the start_instance, adjusted value for number_of_instances). The Management Client can detect the total number of instances with an own request, i.e. start_instance and number of instances each set to ‘0’ in the request, refer to case (d) in the table below (2.20.3.3).

The responsibility and control of this lies entirely with the Management Client; there are no requirements towards the Management Server.

See also 2.20.3.3 for error and exception handling.

2.20.3.3 Error and exception handling

Management Server

The following table gives an overview about the behaviour of the Management Server in the normal cases (a), (b) as well as in various exception cases. The left columns specify the possible combinations of the test_info in the A_NetworkParameter_Read-PDU of NM_ObjectIndex_Read. The right 4 columns specify how the Management Server shall answer with test_info and test_result in the A_NetworkParameter_Response-PDU of NM_ObjectIndex_Read.

Table 3 – Management Server side error handling

case	test_info in Read-PDU			test_result in Response-PDU			Remark
	start_ instance	number_of_ instances		start_ instance	number_of_ instances	Object Index(es)	
(a)	1	> 0		1	as requested in Read_PDU	As many indexes as requested with number_of_- instances	This is the most typical case.
(b)	> 1	> 0		as requested in Read_PDU	as requested in Read_PDU		This allows reading with an offset, in case more instances of the Object Type are available in the Management Server than what can be reported in a single A_NetworkParameter_Response-PDU.
(c)	> 0	0		as requested in Read_PDU	= number of delivered Object Indexes	All indexes starting from start_instance	This is a "lazy client" style: the Management Client does not specify the number of instances it wants, either because it expects that the number will fit in an L_Data_Standard frame, or expects that it can handle a possible L_Data_Extended frame, or is ready to send more requests if the Server would signal the availability of potential other instances.
(d)	0	0		0	current total_number_ of_instances	none	This is a standard exception allowing the Client to firstly detect the total number of instances and then have a systematic discovery without any of the risks of case (c).
(e)	0	x		0	0		This is an error by the Management Client: 0 is not a valid start_instance. The Server does not give information in the response but possibly adjusts the number_of_instances in the response.
(f)	> current total_number_ of_instances	x		0	0		This is an error by the Management Client, which it may make if it has not discovered the total number of instances: the start_index has a value larger than the number of instances.
(g)	> 0	requested number exceeds highest instance		as requested in Read_PDU	= number of delivered object indexes	indexes starting from start_instance until last instance	The Management Client attempts to read more instances than what is available. The Management Server returns all instances it has and corrects the number_of_instances.
(h)	> 0	> number fitting in response frame		as requested in Read_PDU	= number of delivered object indexes, as many as fit in the frame	Indexes starting from start_instance, as many as fit in the frame	The Management Client attempts to read more instances than what the Management Server can hold in its response frame. The Management Server returns as many as possible and corrects the number_of_instances.
(i)	> 0	0, but total number of instances > number fitting in response frame		as requested in Read_PDU	= number of delivered object indexes	Indexes starting from start_instance, as many as fit in the frame	This is a combination of (c) and (h): again, the Management Server returns the maximal set of instances and corrects the number_of_instances.

Management Client

If no response is received, the Management Client shall wait until the time-out has expired. This time-out shall be 3 s [5)]. If no response is received, the Management Client shall assume that the requested Object Type is not present in the Management Server device.

NOTE 2 It is assumed here that a Management Client uses the NM_ObjectIndex_Read-Procedure only for devices supporting the NM_ObjectIndex_Read-Procedure as Management Servers. The support of NM_ObjectIndex_Read by Management Servers shall be defined in the corresponding device Profiles, see also [08].

2.21 NM_SerialNumberDefaultIA_Scan

Use

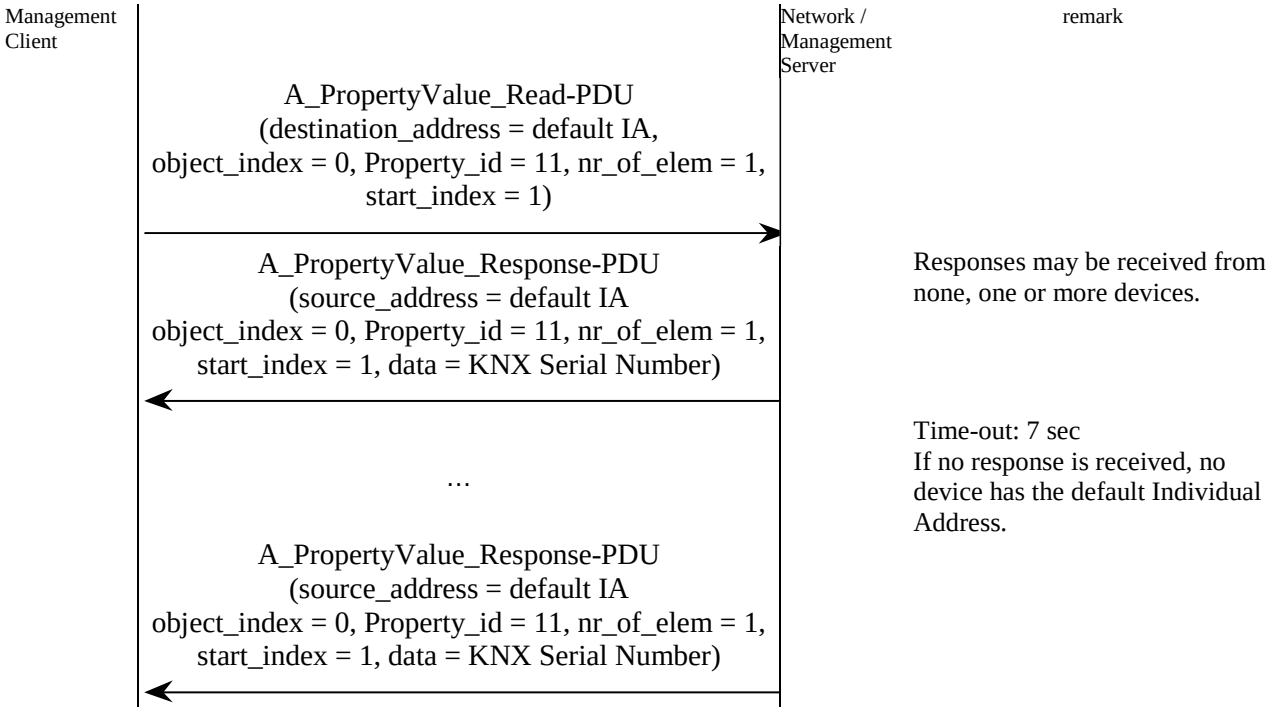
This Network Management Procedure shall be used to obtain the KNX Serial Number of each device of which the Individual Address (IA) is the default Individual Address for the given medium (Subnetwork address as specified in [03], this is, with the Device Address FFh..

Used Application Layer Services for Management

- A_PropertyValue_Read

[5)] 3 s is the Application Layer time-out typically used for confirmed AL-services with response on point-to-point connectionless communication mode.

Sequence



Exception handling

The general exception handling applies.

3 Device Management Procedures

3.1 Introduction

The device Management Procedures describe the rules and procedures for managing a single device. These procedures are device dependent. Therefore a detailed knowledge of the device is required.

The following paragraphs describe the general device Management Procedures. Some parameters depend on the actual BAU-type of the Management Server. The standardized BAU-types can be found in the chapters of [09].

The device Management Procedures can be used to read out the state of a device, write parameters and to load or unload a device with an application.

General Exception handling

In general if an error is detected, the download shall be interrupted and an error-message shall be raised.

If there is another exception handling for a device Management Procedure, this is stated in the description of the procedure.

3.2 DM_Connect

Use

This device Management Procedure shall be used to establish a connection to a Management Server with a specific Individual Address and to check the existence by reading the mask version.

The connection is closed with a DM_Disconnect.

Parameters of the Management Procedure

DM_Connect	(flags)	
flags	bit 0	use connection oriented / connectionless communication 0: connection oriented communication 1: connectionless communication
	All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.	

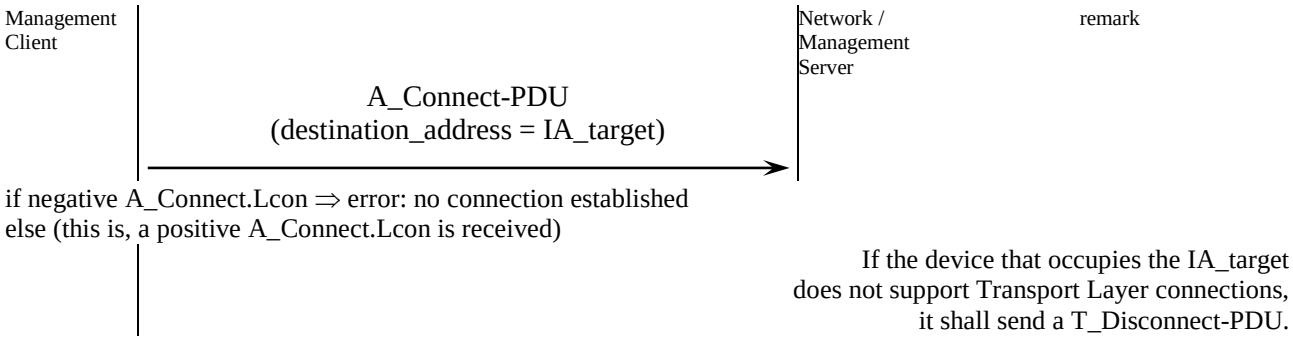
3.2.1 DMP_Connect_RCo

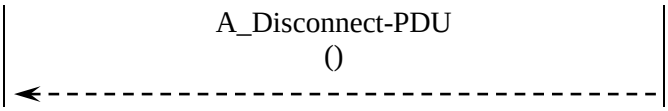
This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Connect
- A_DeviceDescriptor_Read

Sequence

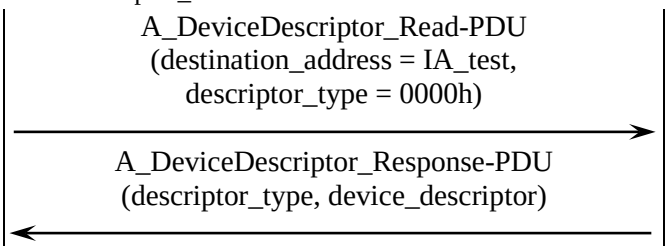




if A_Disconnect-PDU is received ⇒ error: the device refuses the connection-oriented connection
else (no A_Disconnect-PDU is received)

If a device that occupies IA_target is present on the network,
and *does* support Transport Layer connections,
it shall have no other reaction on the bus
than the Layer-2 acknowledge that initiates the above A_Connect.Lcon

The A_DeviceDescriptor_Read-PDU shall use DD0.



If the Management Client receives an A_DeviceDescriptor_Response-PDU it shall conclude that the Transport Layer connection has been established successfully.

If no A_DeviceDescriptor_Response-PDU is received after time-out ⇒ error: no connection established
endif

Exception handling

There are several error situations when building up a connection oriented communication.

- If the T_Connect.req telegram is not acknowledged by a Link Layer acknowledge (negative A_Connect.Lcon) then the Management Server with the Individual Address does not exist or the system is not configured correctly. (e.g. coupler, Domain Address ...)
- If a T_Disconnect-telegram is sent out by the Management Server but no A_DeviceDescriptor_Response, the device with the Individual Address exists. The reason for this may either be that the Management Server is already using another connection, doesn't support connection oriented mode, or the time-out has elapsed.
- If a T_Disconnect-telegram is sent out by the transport layer of the Management Client, then the system may not be configured correctly, or the Management Server doesn't exist.
- If more than one A_DeviceDescriptor_Response-PDU is received, there is more than one device with the target address.

If the above indicates the network may be configured incorrectly, than the network topology shall be checked. In all other cases, the DM_Connect may be repeated several times. If this procedure is not successful, the depending procedures shall be aborted.

3.2.2 Procedure: DMP_Connect_RCl

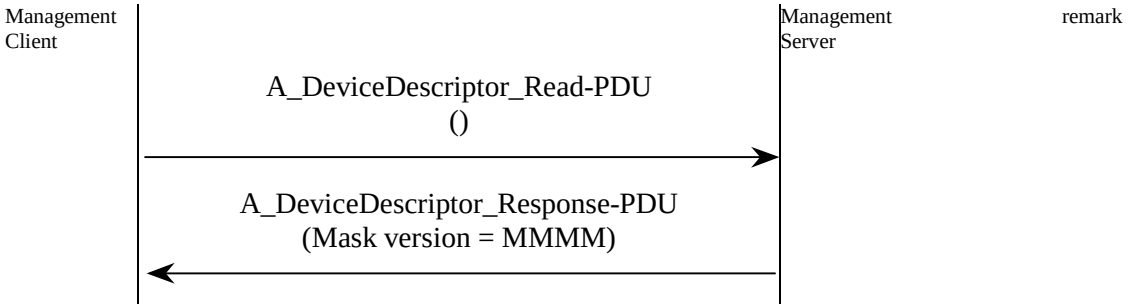
Use

This method uses the connectionless remote communication.

Used Application Layer Services for Management

- A_DeviceDescriptor_Read

Sequence



Exception handling

There are several error situations when building up a connectionless communication.

- If no A_DeviceDescriptor_Response-PDU is received, the Management Server with the Individual Address does not exist or the network is not configured correctly.
- If more than one A_DeviceDescriptor_Response-PDU is received, there is more than one device with the target address.

If the above indicates the network may be configured incorrectly, than the network topology shall be checked. In all other cases, the DM_Connect may be repeated several times. If this procedure is not successful, the depending procedures shall be aborted.

3.2.3 Procedure: DMP_Connect_LEmi1

Use

This Management Procedure shall use the local communication with EMI 1. The Device Descriptor Type 0 shall be read from the memory location 4Eh – 4Fh.

Used EMI-services for Management

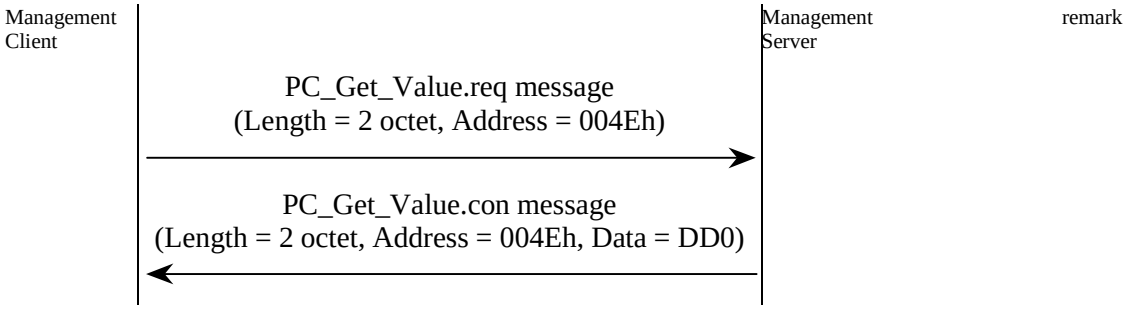
- PC_Get_Value

Parameters of the Management Procedure

DMP_Restart_LEmi1(/* [out] */ DD0, /* [out] */ DmpError)

DD0:	Value of the Device Descriptor 0 as returned by the device.
DmpError:	Possible error indication.

Sequence



Exception handling

The general exception handling is applicable

3.2.5 DMP_Connect_LcEMI

If local management via the cEMI interface is required, the Management Client shall connect to the local device via cEMI and switch the communication mode to cEMI Transport Layer by setting PID_COMM_MODE to “cEMI Transport Layer”. Further discovery (Device Descriptor, manufacturer…) is then done by the classic Application Layer services transferred via cEMI T_Data_Connected and T_Data_Individual.

3.2.6 DMP_Connect_R_KNXnetIPDeviceManagement

The Management Client establishes a KNXnet/IP Device Management connection to the KNX IP - or KNXnet/IP device. The device shall by this autonomously switch its cEMI communication mode to cEMI Transport Layer mode. Further discovery (Device Descriptor, manufacturer…) is then done by the classic Application Layer services transferred via cEMI T_Data_Connected and T_Data_Individual via KNXnet/IP DEVICE_CONFIGURATION_REQUEST frames.

3.2.7 Procedure: DM_DeviceDescriptor_InfoReport

Use

This Management Procedure shall be used to spontaneously send a Device Descriptor value. This procedure is typically spontaneously executed by the Manegement Server (device) and not by the Management Client!

On KNX RF, the A_DeviceDescriptor_InfoReport-PDU shall betransmitted in a RF AddrExtensionType = 0; the frame shall then contain the KNX Serial Number of the sender.

Used Application Layer Services for Management

- A_DeviceDescriptor_InfoReport

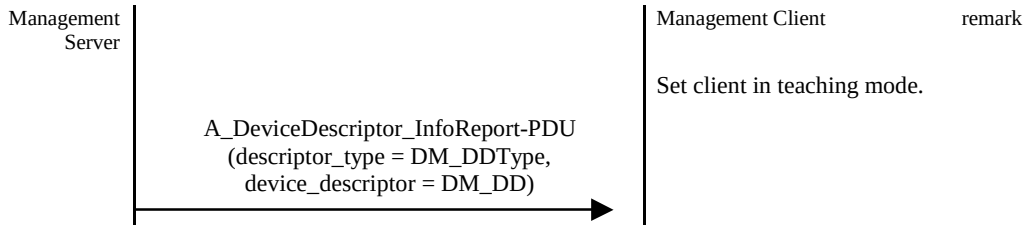
Parameters of the Management Procedure

DM_DeviceDescriptor_InfoReport (/* [in] */ DM_DDType, /* [in] */ DM_DD)

DM_DDType	type of the Device Descriptor
DM_DD	the Device Descriptor of the device

This Management Procedure shall use the system broadcast communication mode.

Sequence



Error and exception handling

Not applicable.

3.3 DM_Disconnect

3.3.1 Use

This device Management Procedure shall be used to close a connection to the Management Server, which was built up with DM_Connect.

A DM_Connect shall be executed before executing this Management Procedure.

DM_Disconnect (flags)

flags	All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

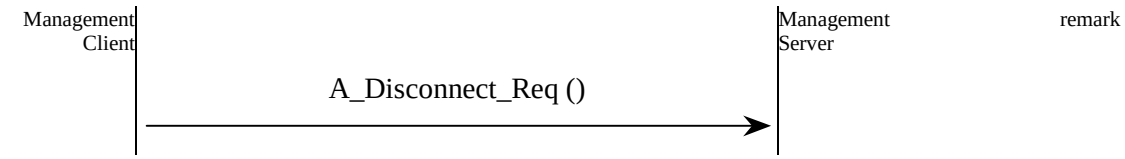
3.3.2 Procedure: DMP_Disconnect_RCo

This method shall use the connection oriented remote communication.

Used Application Layer Services for Management

- A_Disconnect

Sequence



Exception handling

The general exception handling shall apply.

3.3.4 Procedure: DMP_Disconnect_LEmi1

Use

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

None

Parameters of the Management Procedure

DMP_Disconnect_LEmi1()

Sequence

None.

Exception handling

The general exception handling shall apply.

3.4 DM_Identify

3.4.1 General

This Management Procedure shall be used to retrieve data from the Management Server in order to be able to uniquely identify this Management Server and differentiate it from other Management Servers.

3.4.2 DM_Identify_R

Use

This Device Management Procedure shall be used to identify devices with the value 0300h for Device Descriptor type 0. This method shall point-to-point connection oriented - or point-to-point connectionless remote communication mode.

This procedure shall use a *two-step device discovery*.

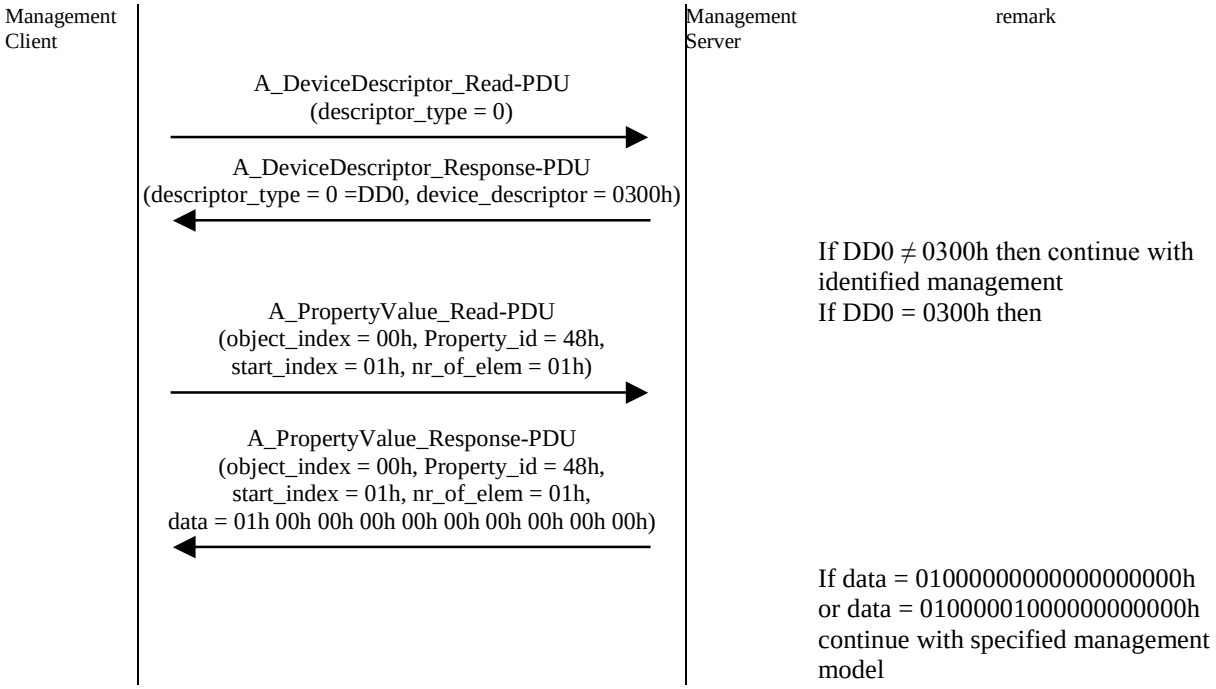
Step 1: Read Device Descriptor Type 0

Step 2: Read PID_MGT_DESCRIPTOR_01

Used Application Layer Services for Management

- A_DeviceDescriptor_Read
- A_PropertyValue_Read

Sequence



3.4.3 DM_Identify_RCo2

This Management Procedure shall use the point-to-point connection-oriented communication mode.

DM_Identify_RCo2	(destination_address)
server IA	Individual Address of the Management Server
device descriptor type 0	device descriptor of the Management Server
manufacturer id	identification of the manufacturer of the Management Server
hardware type	hardware type of the Management Server

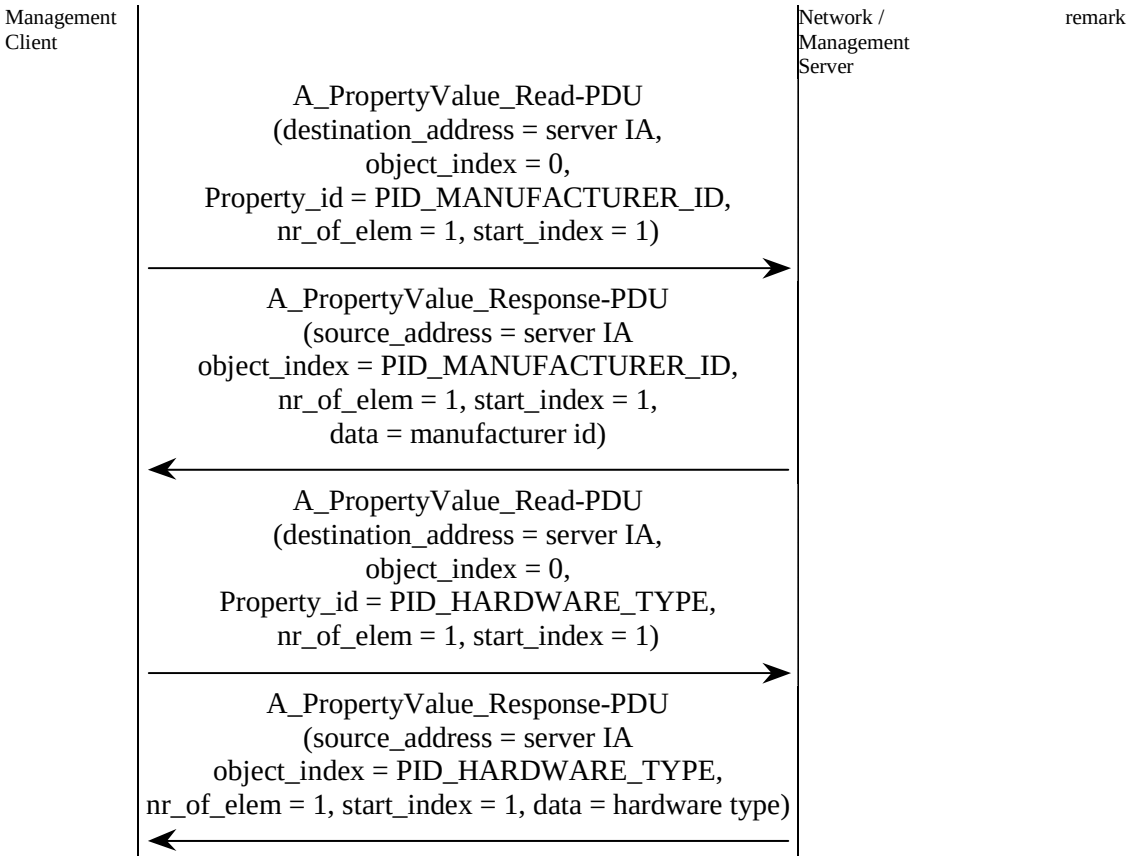
Used Application Layer Services for Management

- A_Connect
- A_DeviceDescriptor_Read
- A_PropertyValue_Read

Sequence

If no Transport Layer connection exists between the Management Server and the Management Client, the Management Client shall execute the procedure DM_Connect_RCo prior to this procedure.

NOTE DM_Connect_RCo already returns the value of the Device Descriptor Type 0 of the Management Server. This result is part of the return of this procedure NM_Identify_RCo2.



Exception handling

The default error handling applies. If any of these services fails (time-out, negative response, no response) then the request shall be repeated up to three times. On further failure, the Management Procedure and the encompassing Configuration Procedure shall be interrupted.

3.5 DM_Authorize

Use

This device Management Procedure shall be used to obtain access authorization. The authorization shall be executed only when it is required by the Management Server.

Whether or not a Management Server supports authorisation can directly be retrieved from the Device Descriptor Type 0 (mask version). In [08] it is specified for which Profiles authorisation is mandatory.

DM_Connect shall be executed before executing this Management Procedure.

DM_Authorize (flags, keys)

flags	All bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
key	key for authorization

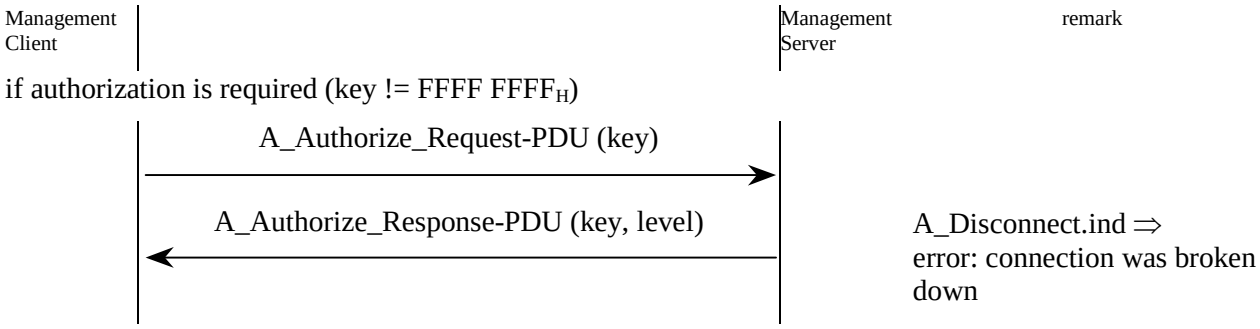
3.5.1 Procedure: DMP_Authorize_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A Authorize

Sequence



Exception handling

The general exception handling shall apply.

3.5.2 DM_Authorize2_RCo

Use

This device Management Procedure DM_Authorize2_RCo shall be used to obtain access authorization. It shall assume that the Management Client has an access key provided by its user. If the Management Client does not have an access key, it shall not be executed.

Opposite to the Management Procedure DM_Authorize_RCo, this Management Procedure DM_Authorize2_RCo does not presume that the device has been locked with the key that is provided to the procedure. Therefore, it authorizes subsequently with the key FFFFFFFFh and with the key client_key and continues with the key that gives the maximal access rights.

NOTE This is the case when the ETS User enters a key to be used to lock the devices and uninitialised devices fresh from the factory are used.

DM_Connect shall be executed before executing this Management Procedure.

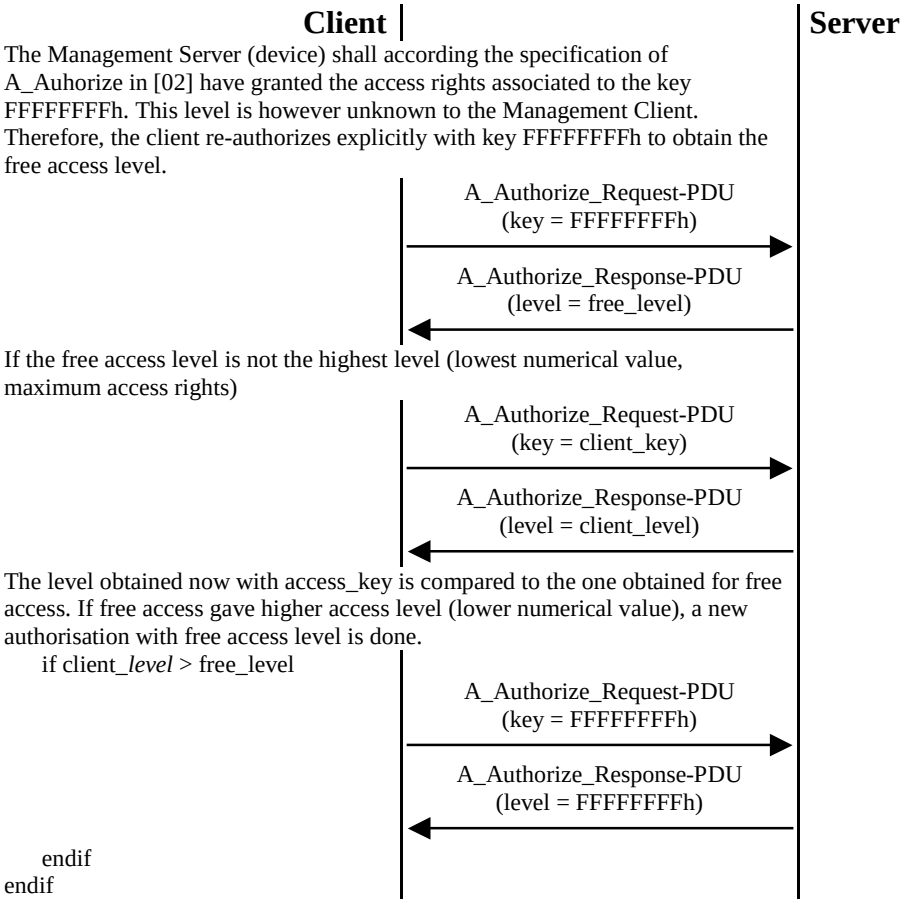
Use

- Profiles System 2
  BIM M112
- Conditions Write access, i.e. modifying memory- or Property contents
  A key must be available

Used Application Layer Services for Management

- A_Authorize_Request

Sequence



Error and exception handling

Failure of any of the contained Application Layer Services shall lead to failure of the entire Configuration Procedure.

The general exception handling shall apply.

3.6 DM_SetKey

Use

This device Management Procedure shall be used to set the key in the Management Server.

A DM_Connect shall be executed before executing this Management Procedure.

DM_SetKey (flags, keys, level)

flags	All bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
key	key for authorization
level	level for which the key is to be set

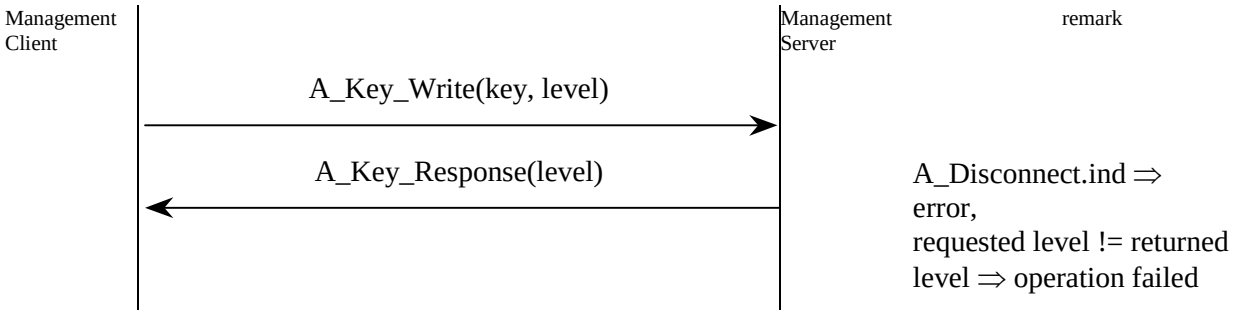
3.6.1 Procedure: DM_SetKey_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Key_Write

Sequence



Exception handling

If the level returned in A_Key_Response is not the same as in the A_Key_Write, the operation was not successful. Possibly an authorization is required.

3.7 DM_Restart

3.7.1 Definition

This Management Procedure can be used in a point-to-point connectionless communication mode (DM_Restart_RCl) or point-to-point connection-oriented communication mode (DM_Restart_RCo).

DM_Restart (flags)

flags	All bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

This Management Procedure shall be used to execute in the Management Server a Basic Restart or a Master Reset.

3.7.1.1 Basic Restart

3.7.1.1.1 Definition

To perform a Basic Restart the Management Server shall

- switch off Programming Mode
- clear runtime errors
- reset all access levels

  This is, the access levels possibly granted to the Management Client shall be set to the highest access level set with FFFFFFFFh.
- switch off safe state
- send an appropriate LM_Reset.ind message through the EMI interface
- reset its KNX communication system

It is recommended to

- break down any Transport Layer connection [6)]

  The Management Server should send a T_Disconnect-PDU to the Management Client.

[6)] Due to the required restart of the communication system, a possible Transport Layer connection will break. It is however recommended that a T_Disconnect-PDU is sent to any communication partner to whom a TL-connection may be established at that time.
Devices have been reported that hold the Transport Layer connection throughout a restart and send a T_Disconnect-PDU when they experience the normal Transport Layer timeout after restart.

3.7.1.1.2 Timing (Management Client and Management Server)

A Management Client may expect a Management Server to successfully have executed a Basic Restart after 1 s and at maximum 5 s. If there is further communication between the Management Client and the restarted Management Server, the reaction failure of successive communication due to longer Management Server restart timing depends on the Configuration Procedure in which this DM_Restart is used.

3.7.1.1.3 Calling a Basic Restart through A_Restart (Management Client and Management Server)

The Basic Restart shall be identified by an A_Restart-PDU

- with the field Response cleared, and
- with all bits 4 to 1 of the APCI/ASDU cleared, and
- with the field restart_type cleared, and
- without the field erase_code, and
- without the field channel_number.

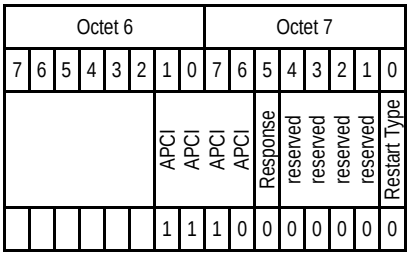


Figure 17 - A_Restart-PDU (example with restart_type = 0)

The Application Layer of the Management Server shall not confirm the A_Restart-service if a Basic Restart is called; to obtain the same result with an AL-confirmation, the Management Client should instead call a Master Reset with Erase Code 00h.

3.7.1.2 Master Reset

3.7.1.2.1 Definition

To perform a Master Reset, the Management Server shall reset its configuration data according the following, if supported and as requested by the Management Client.

- In the Group Address Table and the Group Object Association Table the following link information shall be cleared.
  - If the field Channel Number equals 00h then all link information in these tables shall be cleared.
  - If the field Channel Number differs from 00h, then only the link information in these tables for the Group Objects of the indicated application Channel Number shall be cleared.

  For the error handling concerning invalid values of the field Channel Number, please refer to the specification of the A_Restart-service in [02].
- The application parameters are set to their default value.
  - If the field Channel Number equals 00h then all parameters shall be reset.
  - If the field Channel Number differs from 00h, then only the parameters of the indicated application Channel Number shall be cleared.
  - In case of an E-Mode device with one or more Adjustable E-Mode Channels, this means that the Adjustable Parameter shall for each E-Mode Channel be set back to its default value, which shall be 0. Consequently, other E-Mode Channel configuration data (Channel Codes, Connection Codes and Connection Flags) shall be reset to their default value as well.

- The application is reset to the default application.
- The IA is reset to the devices' medium dependent default IA.
- A Basic Restart is executed.

3.7.1.2.2 Timing (Management Client and Management Server)

To execute the requested Master Reset, the Management Server may need some time, during which it may not be accessible to the Management Client. Therefore, the Management Server shall use the Process Time to report on the time that it needs in worst case for the execution of the requested Master Reset.

This field shall be encoded as a 2 octet unsigned integer value expressed in seconds. This shall comply with the encoding of DPT_TimePeriodSec (DPT_ID = 7.005).

The Management Server user shall fill in the Process Time if this time exceeds 5 s. If not, this field may

- either be left to its default value 0000h, or
- be filled with the correct process time. The MaC is however not required to respect this received process time if this is shorter than the standard process time for the Management Procedure that it is executing. The process time is thus a minimal time for the MaC to wait, not a maximal time.

  EXAMPLE 4 If the MaS indicates it can be available after a Confirmed Restart already after 2 s and the MaS handles a default waiting time of 3 s, then the MaC may still wait for 3 s.

If the Management Server confirms the Master Reset negatively (Error Code ≠ 00h), then it shall set the Process Time to 0000h in the A_Restart_Response-PDU.

If this field is different from 0000h, then the Management Client shall consider it in the error handling (pauses, retries, attempts to reconnect) of its possible further Application Layer services. If the Process Time expires without reaction by the Management Server, then the Management Client **shall** try to call the failed next Application Layer service **one last time** before it may consider the related Configuration Procedure as failed.

The Management Server shall execute the Master Reset only after the A_Restart_Response-PDU has been sent on the bus.

NOTE 3 This shall guarantee that the Master Reset is in all cases a confirmed service: the A_Restart_Response-PDU shall in all cases by sent by the Management Server and received by the Management Client and not go lost due to the possible reset of the communication stack of the Management Server.

NOTE 4 This shall also guarantee that in case the Master Reset changes the Indvidual Address of the Management Server, that the A_Restart_Response-PDU is transmitted on the bus with the current IA of the Management Server prior to calling this service.

3.7.1.2.3 Calling a Master Reset through A_Restart (Management Client and Management Server)

3.7.1.2.3.1 General requirements

The Master Reset shall be identified by an A_Restart-PDU

- with the field Response cleared, and
- with all bits 4 to 1 of the APCI/ASDU cleared, and
- with the field restart_type set to 1, and
- with the field erase_code encoded, and
- with the field Channel Number.

Octet 6								Octet 7								Octet 8								Octet 9							
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	Response	reserved	reserved	reserved	reserved	Restart Type	Erase Code								Channel Number							
						1	1	1	0	0	0	0	0	0	1																

Figure 18 – A_Restart-PDU (example)

The Erase Code shall indicate which Resource value shall be reset in the Management Server. The Channel Number shall indicate to which application channel the reset applies. These fields shall be encoded and used as specified in Table 4.

Table 4 – Definition of Erase Code and Channel Number

Erase Code	Description
01h	**Confirmed Restart** No Resource value shall be reset. This encoding shall allow using the Master Reset as a confirmed alternative to the unconfirmed Basic Restart. Both Erase Code values shall have the same effect. NOTE 5 This encoding uses the same values for the *Erase Code* **as** for the *Reset Command* in [07]. **Channel Number**: Fixed: 00h
02h	**Factory Reset** This shall reset the device to its ex-factory state. Which Resources are reset and their value after reset are implementation dependent. However, for “Factory Reset” and “Factory Reset without IA” there are requirements concerning the effect on system network Resources. See 3.7.1.2.3.2. **Channel Number**: = 00h: The Resources of all Channels shall be reset. ≠ 00h: Only the Resources of the Channel with this given Channel Number shall be reset.
03h	**ResetIA** The IA shall be reset to the medium specific default IA when this A_Restart is executed. **Channel Number**: Fixed: 00h
04h	**ResetAP** Application Program Memory shall be reset to the default application when an A_Restart-PDU is received. **Channel Number**: Fixed: 00h
05h	**ResetParam** Application Parameter Memory shall be reset to its default value when an A_Restart-PDU is received. **Channel Number**: = 00h: The application parameters of all Channels shall be reset. ≠ 00h: The application parameters of only this given Channel shall be reset.

Erase Code	Description
06h	**ResetLinks** Link information for Group Objects (Group Address Table, Group Object Association Table) shall be reset to default state when an A_Restart-PDU is received. **Channel Number**: = 00h: The link information of all Channels shall be reset. ≠ 00h: The link information of only this given Channel shall be reset.
07h	**Factory Reset without IA** This shall reset the device to its ex-factory state. Which Resources are reset and their value after reset are implementation dependent. Opposite to Erase Code 02h, the Individual Address shall not be reset. Further specific requirements as specified in 3.7.1.2.3.2 shall apply. **Channel Number**: = 00h: The Resources of all Channels shall be reset. ≠ 00h: Only the Resources of the Channel with this given Channel Number shall be reset.
00h 08h to FFh	These values are reserved. The Management Client shall not use these Erase Codes. The Management Server shall on reception of an A_Restart-PDU with an Erase Code value in this range - neither execute a Basic Restart nor any Master Reset, and - respond with an A_Restart.res with Error Code “Unsupported Erase Code”. **Channel Number:** not defined

The Management Server shall handle the requested Master Reset as specified above and shall respond with an A_Restart_Response-PDU as follows:

- with the field Response set to 1, and
- with all bits 4 to 1 of the APCI/ASDU cleared, and
- with the field restart_type set to 1, and
- with the field Error Code, and
- with the field Process Time.

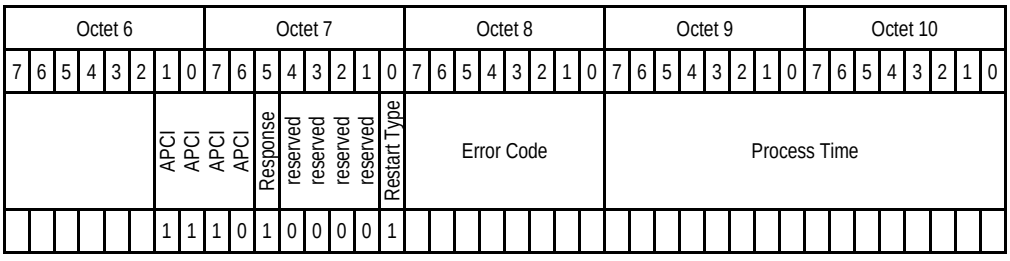


Figure 19 - A_Restart_Response-PDU (example)

The Error Code shall indicate to the Management Client the result of the requested A_Restart from the Management Server. The field Error Code shall be encoded and sent as specified in Table 5.

Table 5 – Error Code

Error Code	Description
00h:	**No Error** The Management Server has properly received the A_Restart-PDU and will execute it.
01h	**Access denied** The Management Server has properly received A_Restart-PDU. Master Reset functionality is in this Management Server protected by authorization and the requesting Management Client is not properly authorized. It is recommended that this Error Code be responded only if Resources are involved that are protected by an access level of 2 and higher.
02h	**Unsupported Erase Code** The Management Server has properly received the A_Restart-PDU; the Management Client may have the required authorization; the requested Erase Code is not supported.
03h	**Invalid Channel Number** This Erase Code value shall be responded in the following cases. - The requested Erase Code requires the Channel Number to be 00h but it is not. - A Channel Number different from 00h is requested but the Management Server does not support application channels. - The Channel Number is used for an application channel that is not supported.
04h to 255h	These values are reserved. The Management Server shall not use any value in this range.

3.7.1.2.3.2 Specific requirements

Factory Reset and Factory Reset without IA

Some system network Resources shall or shall not be reset, as specified in Table 6.

Table 6 – Reset of network Resources in function of the Erase Code

	Erase Code	
	Factory Reset	**Factory Reset without IA**
Parameter	**02h**	**07h**
• IA	shall be reset	shall NOT be reset
• Domain Address (RF and PL)		
• IP address		
• IP address mask		
• IP multicast address		
• IP address assignment method		
• IP default gateway		
• KNXnet/IP Tunnelling PID_ADDITIONAL_-INDIVIDUAL_ADDRESSES		

After a Factory Reset the IP device obviously assumes an IP address that is implementation specific. It may be a default fixed IP address, or it may depend on the default assignment method.

3.7.2 Procedure: DM_Restart_RCl

Use

This method shall use the point-to-point connectionless communication mode.

The Management Client shall prior to calling this Management Procedure with a Master Reset verify that this feature is effectively supported by the Management Server. If not, the procedure shall only be called with a Basic Restart [7].

Used Application Layer services for Management

- A_Restart

Parameters of the Management Procedure

DM_Restart_RCl(/* [in] */ mpp_RestartType, /* [in] */ mpp_EraseCode,
/* [in] */ mpp_ChannelNumber, /* [out] */ mpp_ErrorCode,
/* [out] */ mpp_ProcessTime)

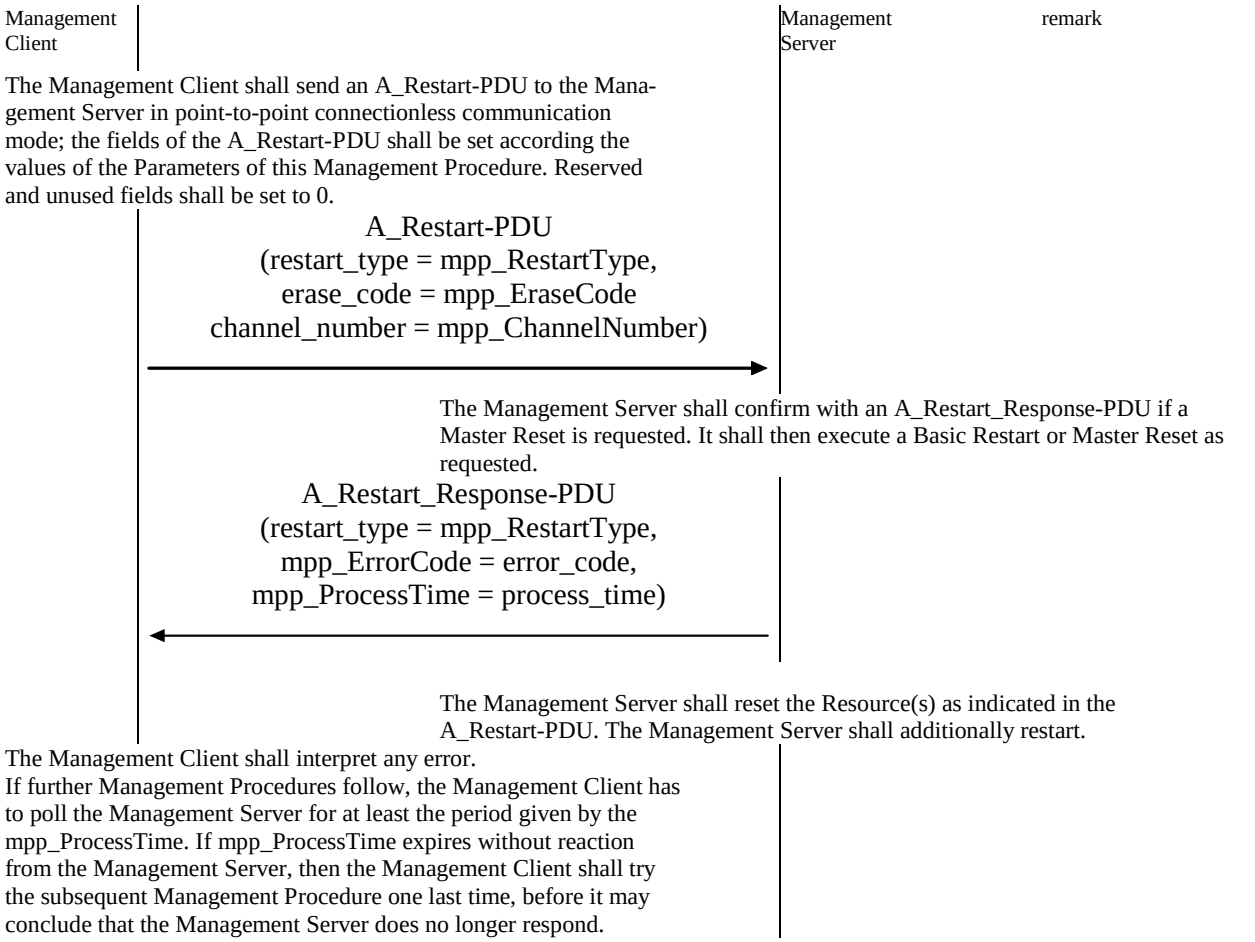
mpp_RestartType:	This Management Procedure Parameter shall indicate whether a Basic Restart or a Master Reset shall be executed.
mpp_EraseCode:	This Management Procedure Parameter shall indicate which Resource(s) shall be reset to its (their) default value. This is void in case only a Basic Restart is executed.
mpp_ChannelNumber:	The number of the application channel that shall be reset or 00h.
mpp_ErrorCode	This Management Procedure Parameter shall contain the Error Code returned by the Management Client.
mpp_ProcessTime	This Management Procedure Parameter shall return to the Management Client the Process Time needed by the Management Server. The Management Client shall consider this mpp_ProcessTime as time-out after which communication attempts following a Master Reset shall be considered without success.

Variables

None.

7) Existing implementations may not check bit 0 of octet 7 and may not react as expected. They may ignore the service entirely, only perform a Basic Restart if a Master Reset is called or exhibit another behaviour.

Sequence



Exception handling

The general exception handling shall be applicable.

The following errors and exceptions shall be checked in the below given priority.

(1) If the Management Server receives an A_Restart-PDU with a reserved field with a value different from 0, then this service request shall be ignored.

(2) If the Management Server receives an A_Restart-PDU but the Management Client does not have the required access rights (KNX Authorization) then it shall respond with an A_Restart_Response-PDU with Error Code = 01h “Access Denied”.

(3) If the Management Server receives an A_Restart-PDU with an Erase Code that it does not support or that is specified as “reserved” then it shall respond with an A_Restart_Response-PDU with Error Code = 02h “Unsupported Erase Code”.

(4) The Management Server shall respond with the Error Code 03h = “Invalid Channel Number” in any of the following cases.
- It receives an A_Restart-PDU with a Channel Number that is not 00h with an Erase Code for which the Channel Number shall be 00h.
- It receives an A_Restart-PDU with an Erase Code that allows the Channel Number to be different from 00h; the Channel Number is different from 00h but the Management Server does not support application channels.
- It receives an A_Restart-PDU with an Erase Code that allows the Channel Number to be different from 00h; the Channel Number is different from 00h and the Management Server does support application channels but does not have a channel with the requested channel number.

3.7.3 Procedure: DM_Restart_RCo

Use

This method shall use the point-to-point connection oriented remote communication.

The Management Client shall prior to calling this Management Procedure with a Master Reset verify that this feature is effectively supported by the Management Server. If not, the procedure shall only be called with a Basic Restart [8].

A DM_Connect shall be executed before executing this Management Procedure.

After reception of the A_Restart-PDU this Transport Layer connection breaks down with the execution of the A_Restart service; nevertheless an explicit DM_Disconnect procedure shall follow.

Used Application Layer services for Management

- A_Restart

Parameters of the Management Procedure

DM_Restart_RCo (/* [in] */ mpp_RestartType, /* [in] */ mpp_EraseCode,
/* [in] */ mpp_ChannelNumber, /* [out] *mpp_ErrorCode,
/* [out] */ mpp_ProcessTime)

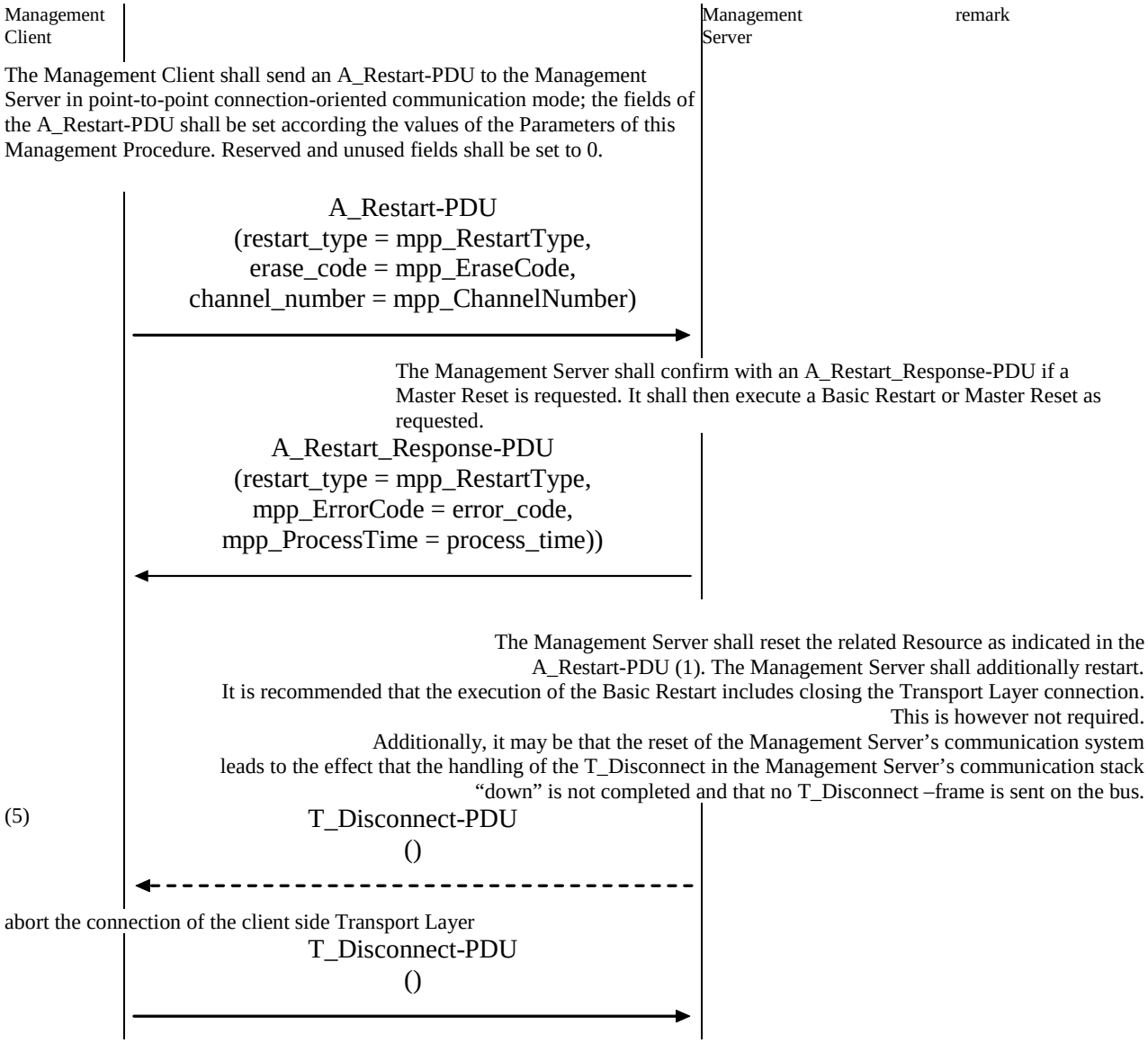
mpp_RestartType:	This Management Procedure Parameter shall indicate whether a Basic Restart or a Master Reset shall be executed.
mpp_EraseCode:	This Management Procedure Parameter shall indicate which Resource(s) shall be reset to its (their) default value. This is void in case only a Basic Restart is executed.
mpp_ChannelNumber:	The number of the application channel that shall be reset or 00h.
mpp_ErrorCode	This Management Procedure Parameter shall contain the Error Code returned by the Management Client.
mpp_ProcessTime	This Management Procedure Parameter shall return to the Management Client the Process Time needed by the Management Server. The Management Client shall consider this mpp_ProcessTime as time-out after which communication attempts following a Master Reset shall be considered without success.

Variables

None.

[8] Existing implementations may not check bit 0 of octet 7 and may not react as expected. They may ignore the service entirely, only perform a Basic Restart if a Master Reset is called or exhibit another behaviour.

Sequence



Exception handling

The general exception handling shall be applicable.

(1) If the Management Server receives an A_Restart-PDU with a reserved field with a value different from 0, then this service request shall be ignored.

(2) If the Management Server receives an A_Restart-PDU but the Management Client does not have the required access rights (KNX Authorization) then it shall respond with an A_Restart_Response-PDU with Error Code = 01h "Access Denied".

(3) If the Management Server receives an A_Restart-PDU with an Erase Code that it does not support then it shall respond with an A_Restart_Response-PDU with Error Code = 02h "Unsupported Erase Code".

(4) The Management Server shall respond with the Error Code 03h = “Invalid Channel Number” in any of the following cases.
- It receives an A_Restart-PDU with a Channel Number that is not 00h with an Erase Code for which the Channel Number.
- It receives an A_Restart-PDU with an Erase Code that allows the Channel Number to be different from 00h; the Channel Number is different from 00h but the Management Server does not support application channels.
- It receives an A_Restart-PDU with an Erase Code that allows the Channel Number to be different from 00h; the Channel Number is different from 00h and the Management Server does support application channels but does not have a channel with the requested channel number.

(5) All telegrams sent out by the Management Server shall be ignored, except negative TL-confirmations.

In particular, the recommended T_Disconnect-PDU from the Management Server may or may not be sent on the bus. The Management Client shall take that into account.

1 Regardless of whether or not a T_Disconnect-PDU is received from the Management Server, the Management Client shall issue a T_Disconnect-PDU to the Management Server.

2 The Management Client shall wait for a possible T_Disconnect-PDU for 6 seconds.(See NOTE 6)
- If after this time no T_Disconnect-PDU is received, then this shall **not** be regarded as a protocol error.
- If the Management Client receives a T_Disconnect-PDU then it shall ignore this message. This T_Disconnect-PDU may be received before or after the “own” mandatory T_Disconnect-PDU from the Management Client specified under 1.

In both cases, at first after this time-out has elapsed, the Management Client shall continue the possible further configuration of the Management Server: the configuration shall not be continued while this time-out has not elapsed.

NOTE 6 This is because a possible subsequent T_Connect-PDU from the MaC and the awaited T_Disconnect-PDU from the MaS may miss each other on the network. Cause and consequence are then unclear to both MaS and MaC, which will leave the TL state machines in an unpredictable state.

3.7.4 Procedure: DMP_Restart_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

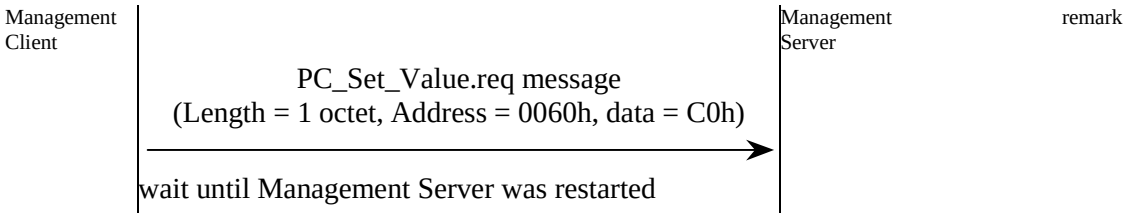
- PC_Set_Value

Parameters of the Management Procedure

DMP_Restart_LEmi1(/* [out] */ DmpError)

DmpError: Possible error indication.

Sequence



Exception handling

The general exception handling shall apply.

3.8 DM_Delay

3.8.1 Use

This device Management Procedure shall be used to wait a specified time before starting the next action.

DM_Delay (flags, delay time)

delay time	Time in milliseconds
flags	All bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

3.8.2 Procedure: DMP_Delay

Used Application Layer Services for Management

None.

Sequence

Management Client		Management Server	remark
	delay for specified time		

Exception handling

The general exception handling shall apply.

3.9 DM_IndividualAddressRead

3.9.1 Use

This device Management Procedure shall be used to read out the Individual Addresses of the local device, independent of the Programming Mode. For remote procedures please refer to the clause 2 "Network Management Procedures".

A DM_Connect shall be executed before executing this Management Procedure.

DM_IndividualAddressRead (Individual Address)

Individual Address	contains the Individual Address of the device

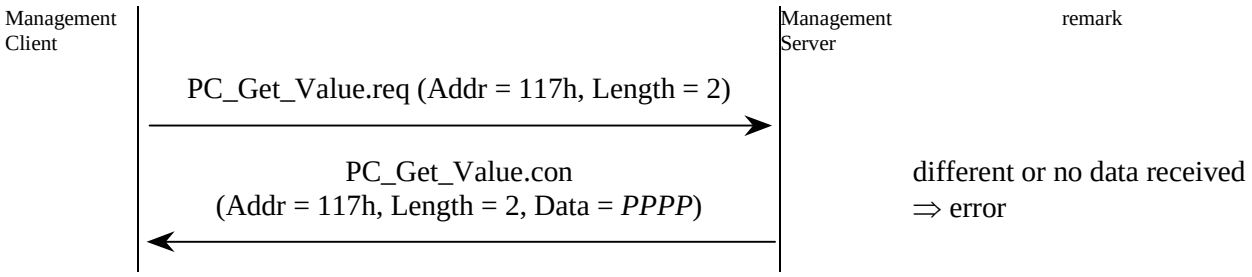
3.9.2 Procedure: DMP_IndividualAddressRead_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

- PC_Get_Value.req

Sequence



Exception handling

The general exception handling shall apply.

3.10 DM_IndividualAddressWrite

3.10.1 Use

This device Management Procedure shall be used to write the Individual Addresses of the local device, independent of the Programming Mode. For remote procedures please refer to the clause 2 "Network Management Procedures".

A DM_Connect shall be executed before executing this Management Procedure.

DM_IndividualAddressWrite (Individual Address)

Individual Address	contains the Individual Address of the device

3.10.2 Procedure: DMP_IndividualAddressWrite_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

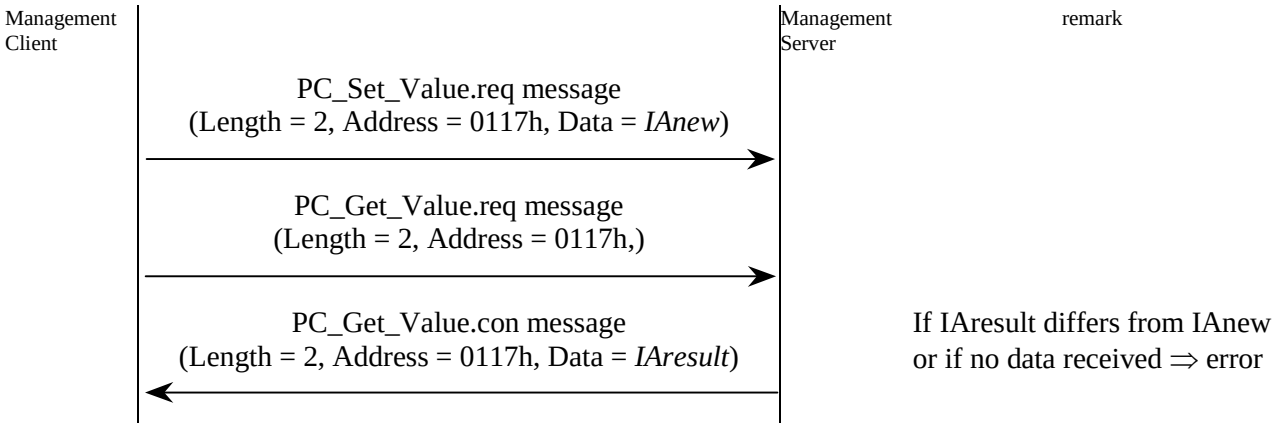
- PC_Get_Value
- PC_Set_Value

Parameters of the Management Procedure

DMP_IndividualAddressWrite_LEmi1(/* [in] */ IAnew, /* [out] */ IAresult)

IAnew:	The Individual Address that shall be written in the device.
IAresult:	The Individual Address as read back from the device after writing IAnew

Sequence



Exception handling

The general exception handling shall apply.

3.11 DM_DomainAddress_Read

3.11.1 Use

This device Management Procedure shall be used to read the Domain Address of the local device, independent of the Programming Mode. For remote procedures please refer to the clause 2 "Network Management Procedures".

A DM_Connect shall be executed before executing this Management Procedure.

DM_DomainAddressRead	(Domain Address)
Domain Address	contains the Domain Address of the device

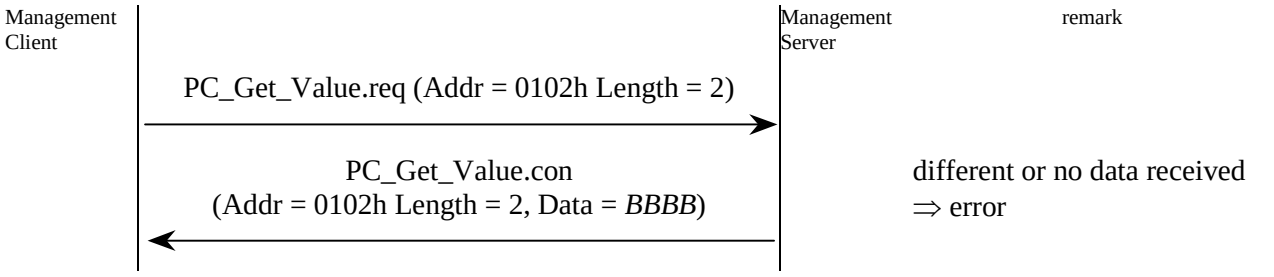
3.11.2 Procedure: DMP_DomainAddressRead_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management for Management

- PC_Get_Value.req

Sequence



Exception handling

The general exception handling shall apply.

3.12 DM_DomainAddressWrite

3.12.1 Use

This device Management Procedure shall be used to write the Domain Address of the local device, independent of the Programming Mode. For remote procedures please refer to the clause 2 "Network Management Procedures".

A DM_Connect shall be executed before executing this Management Procedure.

DM_DomainAddressWrite (Domain Address)

Domain Address	contains the Domain Address of the device

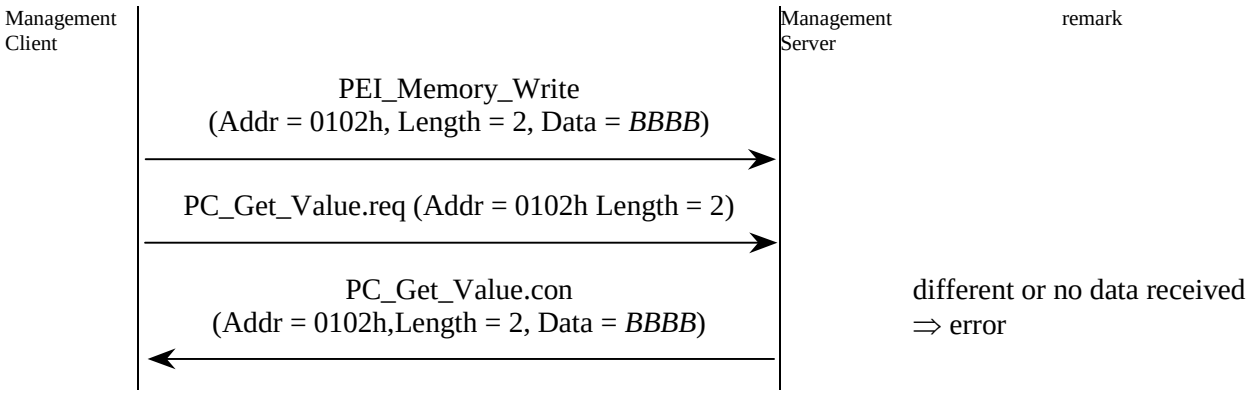
3.12.2 Procedure: DMP_DomainAddressWrite_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

- PC_Get_Value.req
- PEI_Memory_Write

Sequence



Exception handling

The general exception handling shall apply.

3.13 DM_ProgMode_Switch

3.13.1 Use

This device Management Procedure shall switch the Programming Mode of the device.

A DM_Connect shall be executed before executing this Management Procedure.

DM_ProgMode_Switch (flags, mode)

flags	All bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
mode	0: switch Programming Mode off 1: switch Programming Mode on

3.13.2 Procedure: DMP_ProgModeSwitch_RCo

This Management Procedure shall use the connection oriented communication mode.

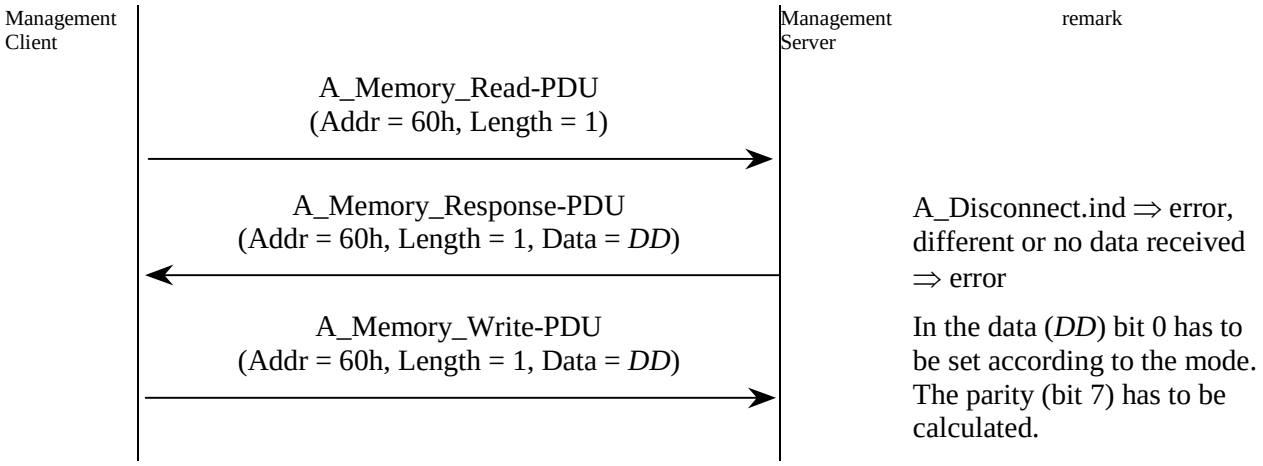
The Programming Mode shall be realised as “Programming Mode – Realisation Type 2” as specified in [03].

NOTE This means that the state of the Programming Mode is located at memory address 60h.

Used Application Layer Services for Management

- A_Memory_Read
- A_Memory_Write

Sequence



Exception handling

The general exception handling shall apply.

3.13.3 Procedure: DMP_ProgModeSwitch_LEmi1

This Management Procedure shall use the local communication with EMI 1.

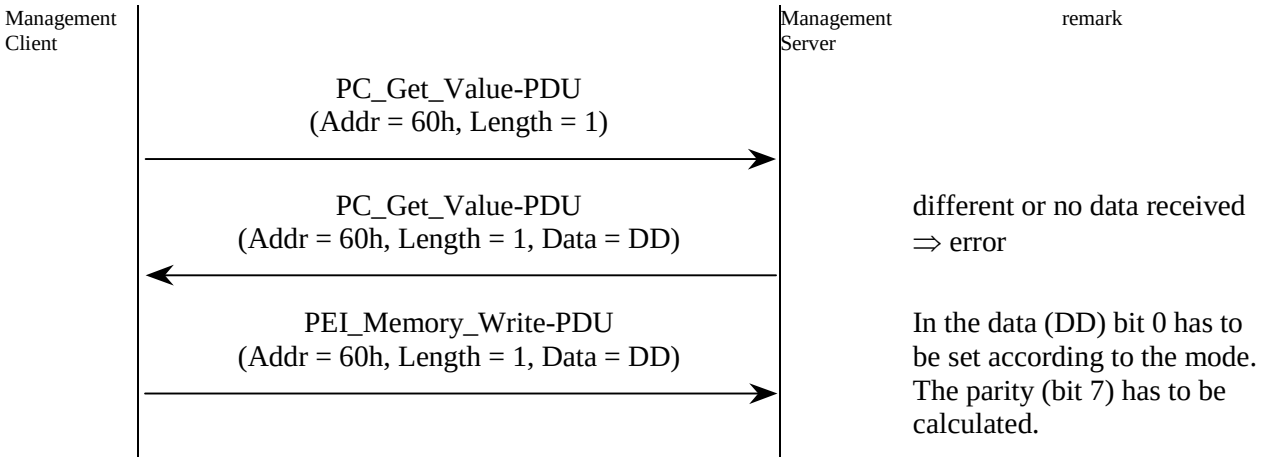
The Programming Mode shall be realised as “Programming Mode – Realisation Type 2” as specified in [03].

NOTE This means that the state of the Programming Mode is located at memory address 60h.

Used EMI-services for Management

- PC_Get_Value.req
- PEI_Memory_Write

Sequence



Exception handling

The general exception handling shall apply.

3.14 DM_PeiTypeVerify

3.14.1 Use

This device Management Procedure shall read the current PEI type of the device and compare it with the specified data. The data shall be located either in the management control or in the data block.

A DM_Connect shall be executed before executing this Management Procedure.

DM_PeiTypeVerify (flags, data)

flags	bit 0	location of data 0: in data block 1: in management control
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress		specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
data		the data that are compared by this Management Procedure. The data can be located in the data block or in the Management Procedure

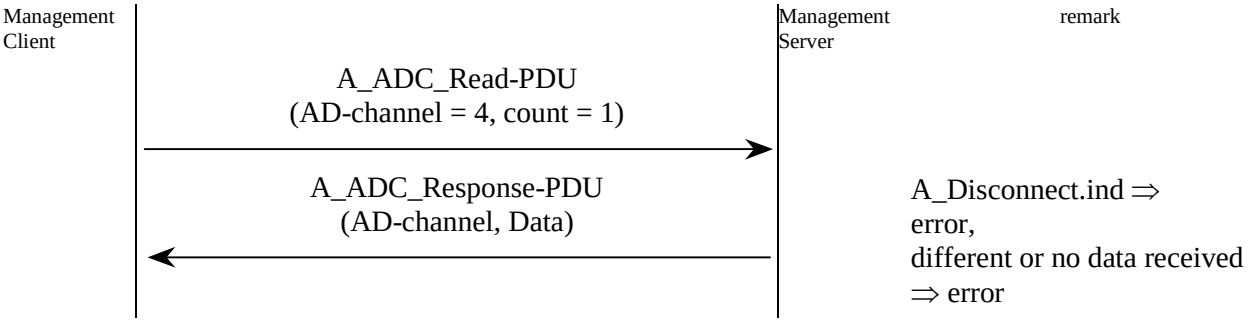
3.14.2 Procedure: DMP_PeiTypeVerify_RCo_ADC

This Management Procedure shall use the connection oriented communication mode.
The value shall be read via the service A_ADC_Read.

Used Application Layer Services for Management

- A_ADC_Read

Sequence



The formula to calculate the PEI type is:

$$PEI\_Type = \frac{10 \cdot ADC\_Value + 60}{128}$$

Exception handling

The general exception handling shall apply.

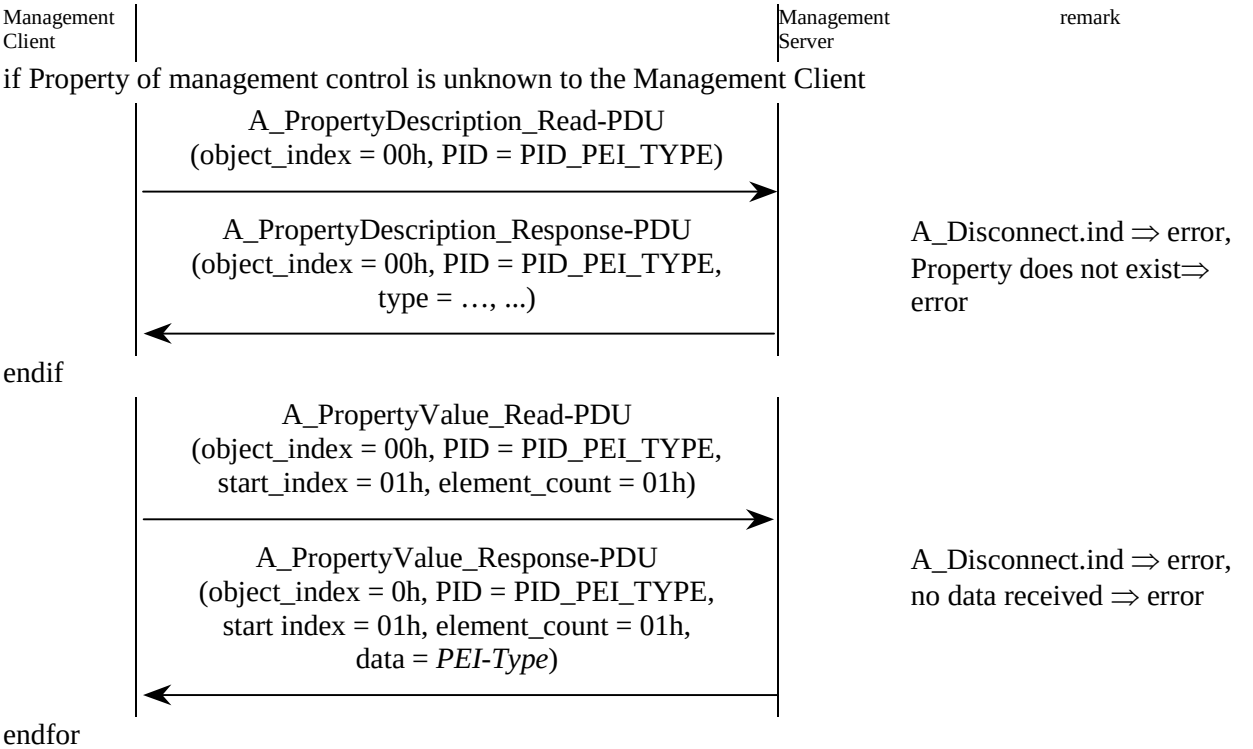
3.14.3 Procedure: DMP_PeiTypeVerify_R_IO

This Management Procedure shall use the connection oriented or connectionless communication mode. The value shall be read via the Interface Objects.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

The general exception handling shall apply.

3.15 DM_PeiTypeRead

3.15.1 Use

This device Management Procedure shall read the current PEI type of the device and store the value in the specified data block.

A DM_Connect shall be executed before executing this Management Procedure.

DM_PeiTypeRead (flags, dataBlockStartAddress, data)

flags	bit 0	location of data 0: in data block 1: -
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress		specifies the address where the data are located in the data block.
data		the data read by this Management Procedure

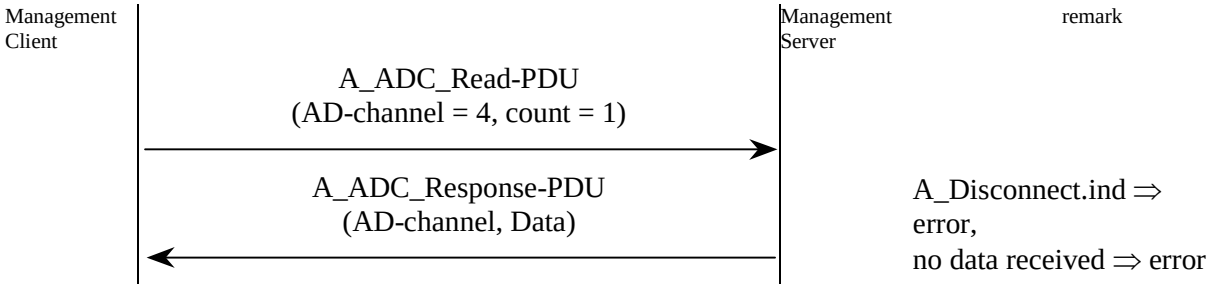
3.15.2 Procedure: DMP_PeiTypeRead_RCo_ADC

This Management Procedure shall use the connection oriented communication mode.
The value shall be read via the service A_ADC_Read.

Used Application Layer Services for Management

- A_ADC_Read

Sequence



The formula to calculate the PEI type is:

$$PEI\_Type = \frac{10 \cdot ADC\_Value + 60}{128}$$

Exception handling

The general exception handling shall apply.

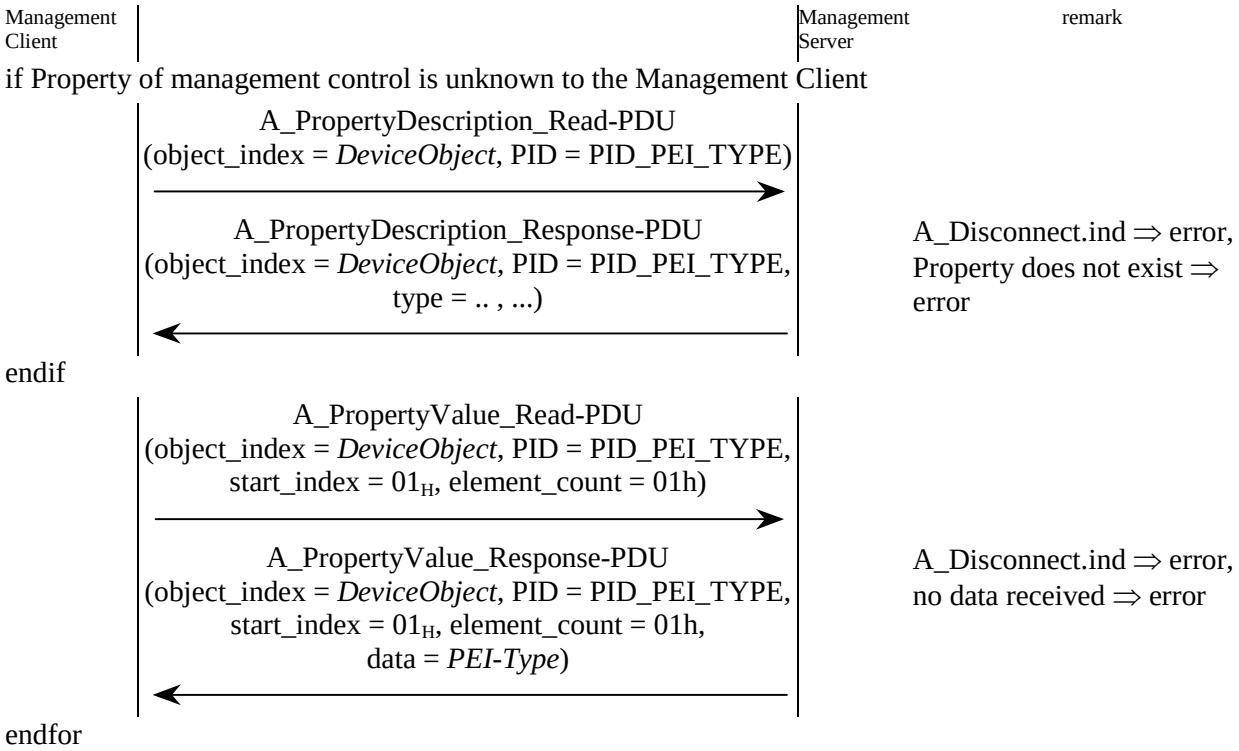
3.15.3 Procedure: DMP_PeiTypeRead_R_IO

This Management Procedure shall use the connection oriented or connectionless communication mode. The value shall be read via the Interface Objects.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

The general exception handling shall apply.

3.16 DM_MemWrite

3.16.1 Use

This device Management Procedure shall write a contiguous block of data to the specified memory addresses.

The data shall be located either in the management control or in the data block. Only the data that is specified in the data block shall be written. Depending on the flag the data shall be verified immediately. If the deviceStartAddress is higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

DM_MemWrite (flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)

flags	bit 0 location of data 0: in data block 1: in Management Procedure bit 1 verify enabled / disabled 0: disabled 1: enabled All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress	address of first memory octet that is written by this Management Procedure
deviceEndAddress	address of the last octet that is written by this Management Procedure
data	the data that are transferred by this Management Procedure. The data can be located in the data block or in the Management Procedure.

Data Format

code	flags	dataBlockStartAddress	deviceStartAddress	deviceEndAddress	reserved / data
20h	FFh	BBBB BBBB	SSSS SSSS	EEEE EEEE	00h / DDh
1 octet	1 octet	4 octet	4 octet	4 octet	18 octet

3.16.2 Procedure: DMP_MemWrite_RCo

Use

This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall not be used.

Preconditions

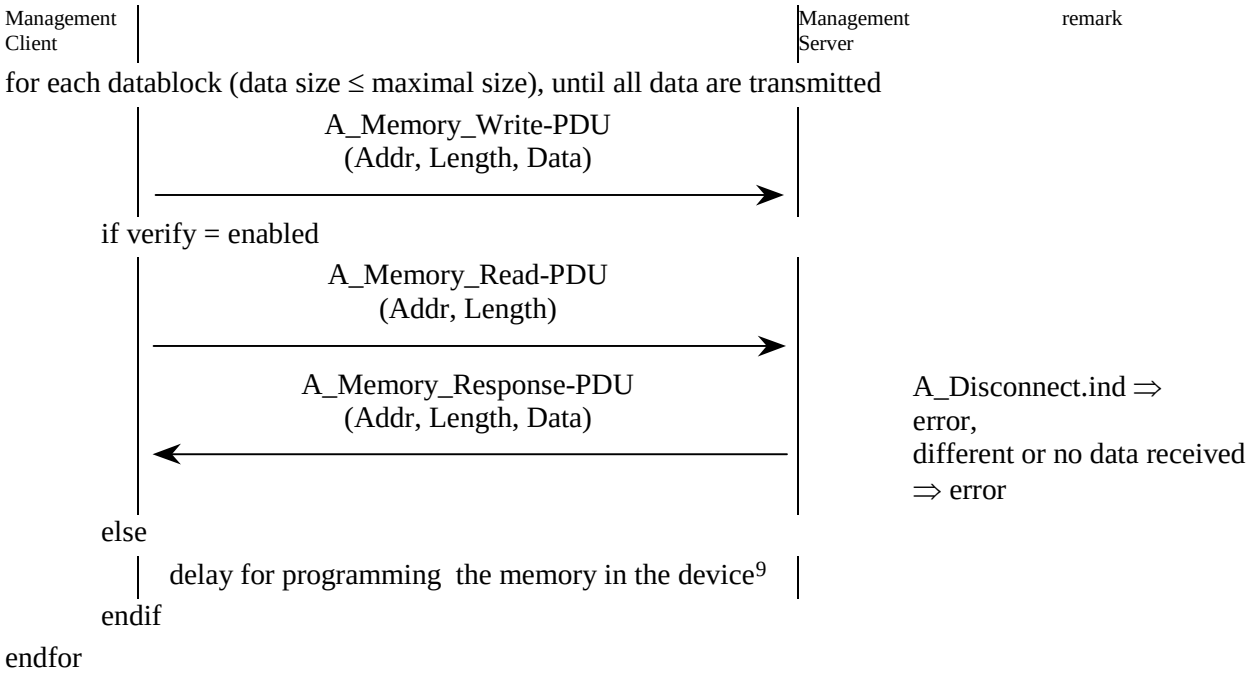
This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_Memory_Read-PDUs and/or A_Memory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 12 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_Memory_Write
- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.16.3 Procedure: DMP_MemWrite_RCoV

This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall be used.

Preconditions

This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_Memory_Read-PDUs and/or A_Memory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

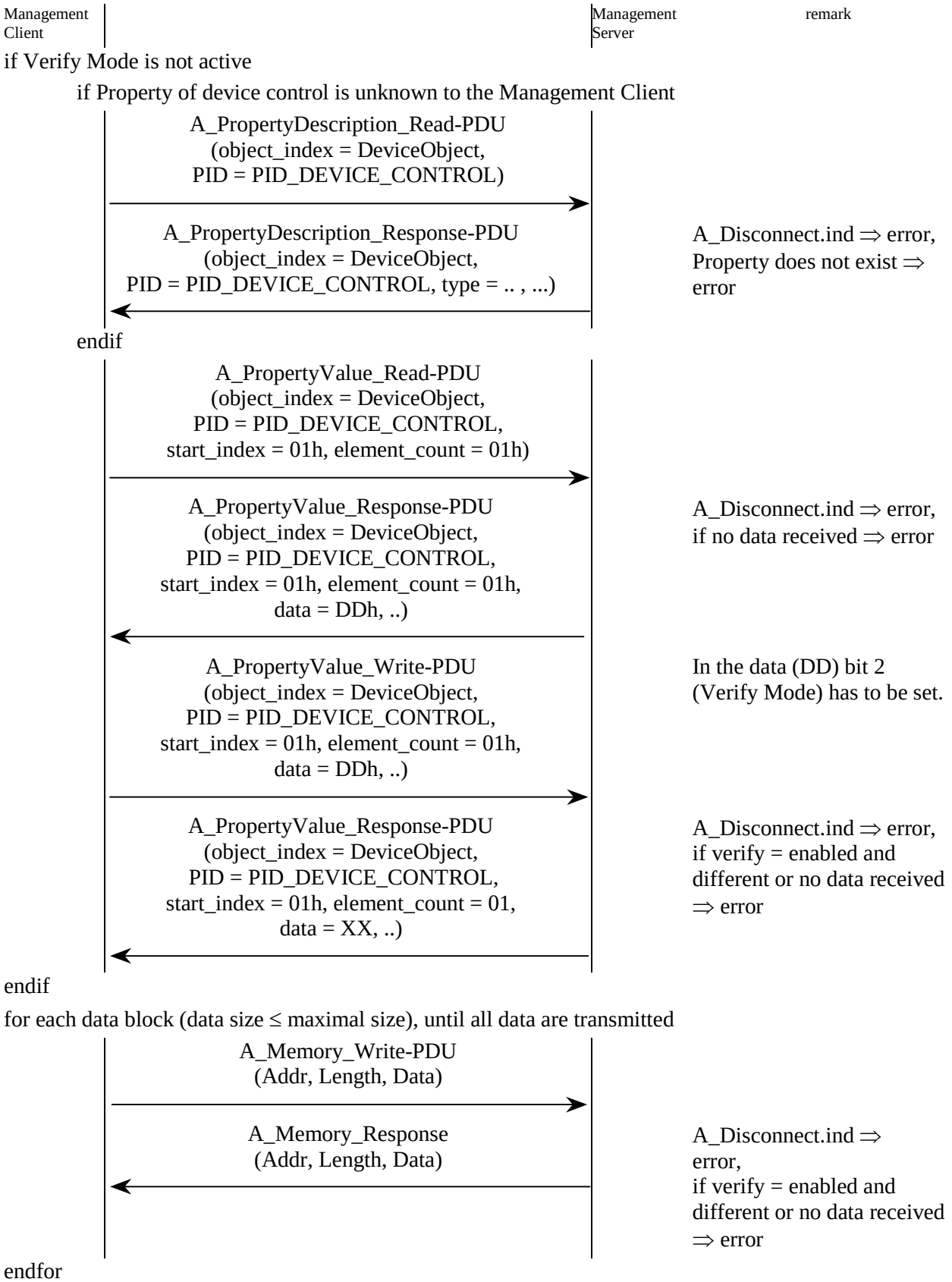
- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 12 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_Memory_Write

9) The delay time depends on the Management Server and on the amount of written octets (see [08]).

Sequence



Exception handling

The general exception handling shall apply.

3.16.4 Procedure: DMP_MemWrite_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

- PC_Get_Value
- PC_Set_Value

Parameters of the Management Procedure

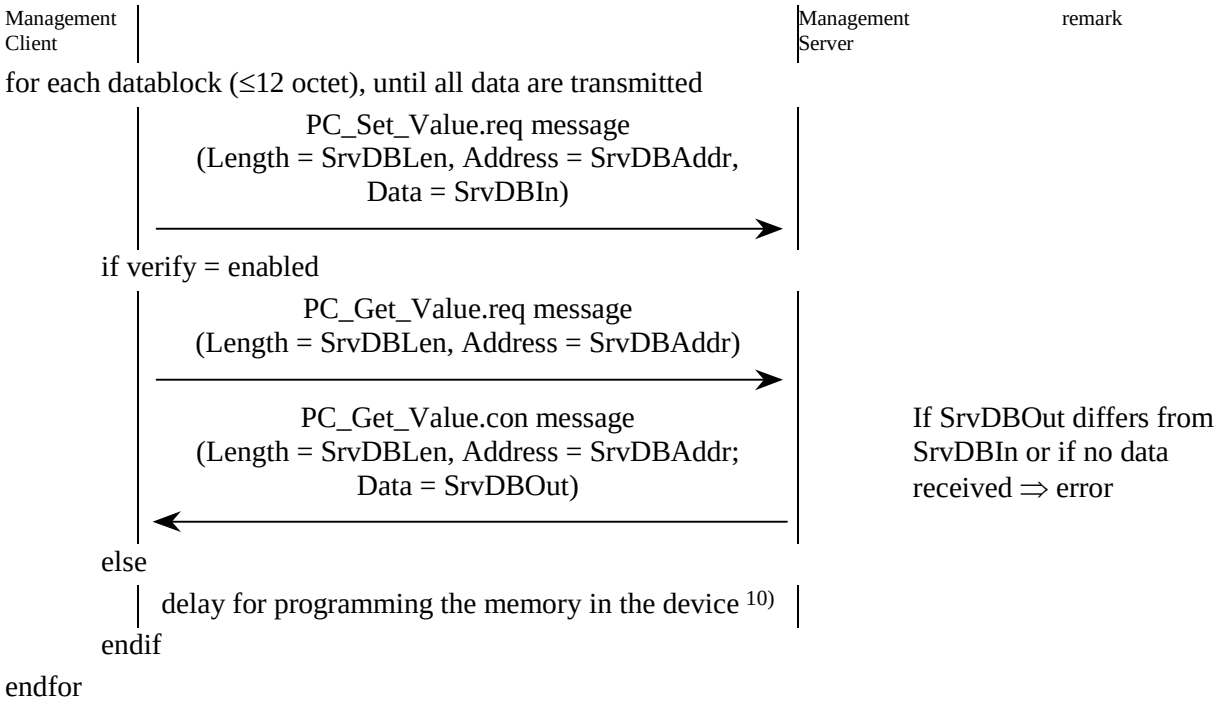
DMP_MemWrite_LEmi1(/* [in] */ DmpStartAddr, /* [in]*/ DmpEndAddr, /* [in]*/ DmpData, /* [out] */ DmpError)

DmpStartAddr:	The start address in the memory of the device into which the Data shall be written.
DmpEndAddr:	The end address in the memory of the device into which the Data shall be written.
DmpData:	The Data to be written in the device.
DmpError:	Possible error indication.

Service parameters

SrvDataIn:	The data that shall be written in the device in one call of the service PC_Set_Value.
SrvDataOut:	The data as read back from the device for each call of the service PC_Set_Value.
SrvDBLen:	The length of the datablock written in one call of the service PC_Set_Value. This shall be 12 octets for all datablocks except for the last one, which may be smaller.
SrvDBAddr:	The start address in the memory of the device where the current datablock shall be written.

Sequence



Exception handling

The general exception handling shall apply.

3.17 DM_MemVerify

3.17.1 Use

This device Management Procedure shall read a contiguous block of memory and compare it with the specified data. The data shall be located either in the management control or in the data block. Only the data that are specified in the data block shall be compared. If the deviceStartAddress is higher than the deviceEndAddress this Management Procedure is skipped.

A DM_Connect shall be executed before executing this Management Procedure.

10) The delay time shall depend on the Management Server and on the amount of written octets (see [08]).

DM_MemVerify		(flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)
flags	bit 0	location of data 0: in data block 1: in management control
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress		specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress		address of first memory octet that is compared by this Management Procedure
deviceEndAddress		address of the last octet that is compared by this Management Procedure
data		the data that are compared by this Management Procedure. The data can be located in the data block or in the Management Procedure.

3.17.2 Procedure: DMP_MemVerify_RCo

This Management Procedure shall use the connection oriented communication mode.

Preconditions

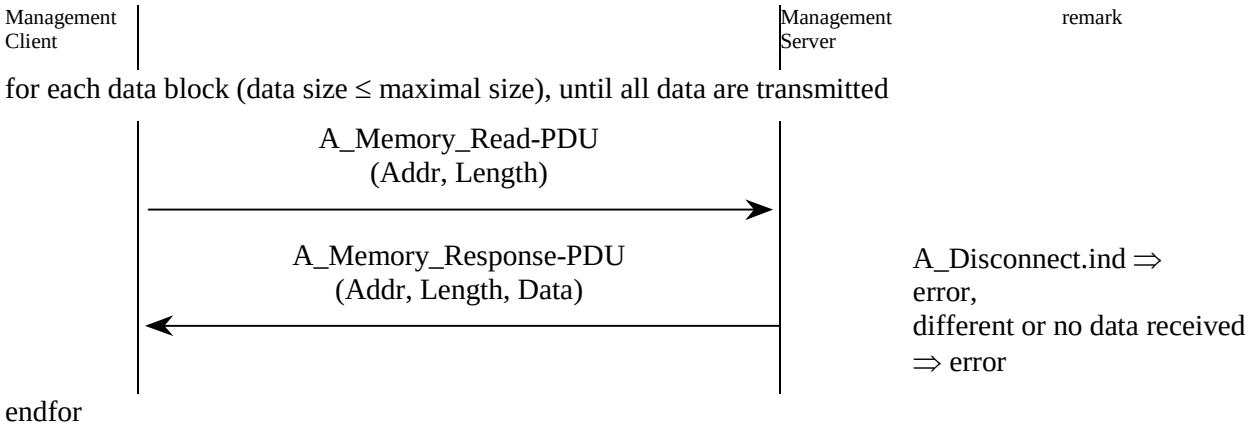
This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_Memory_Read-PDUs and/or A_Memory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 12 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

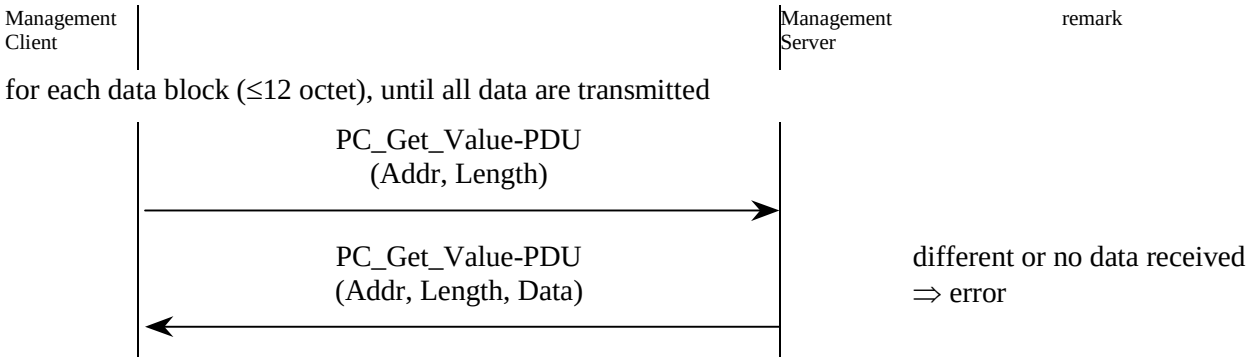
3.17.3 Procedure: DMP_MemVerify_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used Application Layer Services for Management

- PC_Get_Value

Sequence



Exception handling

The general exception handling shall apply.

3.18 DM_MemRead

3.18.1 Use

This device Management Procedure shall read a contiguous block of memory and store it in the data block. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

DM_MemRead (flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)

flags	bit 0 location of data 0: in data block 1: - All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0
deviceStartAddress	address of first memory octet that is read by this Management Procedure
deviceEndAddress	address of the last octet that is read by this Management Procedure
data	the data that are read by this Management Procedure. The data are stored in the data block.

3.18.2 Procedure: DMP_MemRead_RCo

This Management Procedure shall use the connection oriented communication mode.

Preconditions

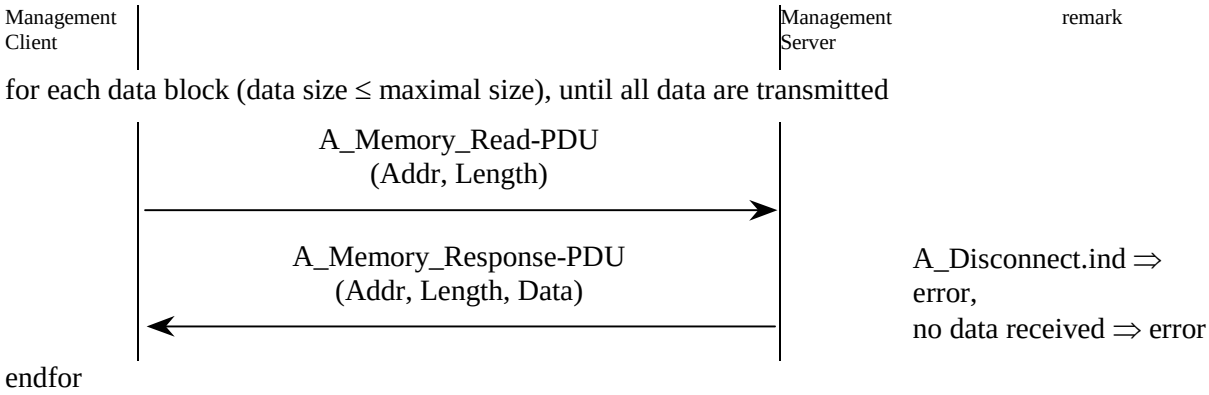
This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_Memory_Read-PDUs and/or A_Memory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 12 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.18.3 Procedure: DMP_MemRead_LEmi1

This Management Procedure shall use the local communication with EMI 1.

Used EMI-services for Management

- PC_Get_Value

Parameters of the Management Procedure

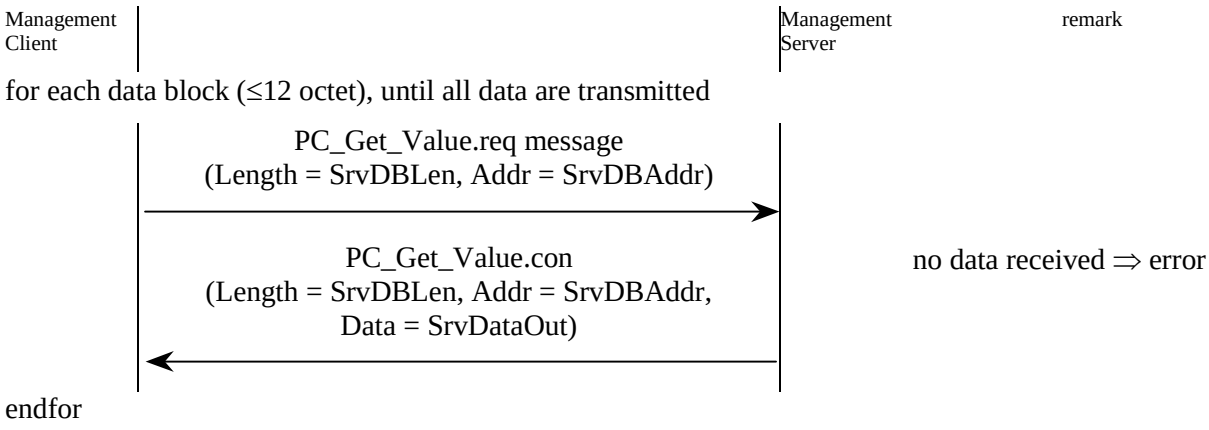
DMP_MemRead_LEmi1(/* [in] */ DmpStartAddr, /* [in] */ DmpEndAddr, /* [out] */ DmpData, /* [out] */ DmpError)

DmpStartAddress:	The start address of the memory of the device from which the Data shall be read.
DmpEndAddress:	The end address of the memory of the device from which the Data shall be read.
DmpData:	The contents of the memory as returned by the device.
DmpError:	Possible error indication.

Service parameters

SrvDataOut:	The data as read from the device for each call of the service PC_Get_Value.
SrvDBLen:	The length of the datablock that shall be in one call of the service PC_Get_Value. This shall be 12 octets for all datablocks except for the last one, which may be smaller.
SrvDBAddr:	The start address in the memory of the device from which the current datablock shall be read.

Sequence



Exception handling

The general exception handling shall apply.

3.19 DM_UserMemWrite

3.19.1 Use

This device Management Procedure shall write a contiguous block of data to the specified memory addresses of the user memory in the Management Server. The data shall be located either in the management control or in the data block. Only the data that are specified in the data block shall be written. Depending on the flag the data shall be verified immediately. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

DM_UserMemWrite		(flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)
flags	bit 0	location of data 0: in data block 1: in Management Procedure
	bit 1	verify enabled / disabled 0: disabled 1: enabled
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress		specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress		address of first user memory octet that is written by this Management Procedure
deviceEndAddress		address of the last user memory octet that is written by this Management Procedure
data		the data that are transferred by this Management Procedure. The data can be located in the data block or in the Management Procedure.

3.19.2 Procedure: DMP_UserMemWrite_RCo

This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall not be used.

Preconditions

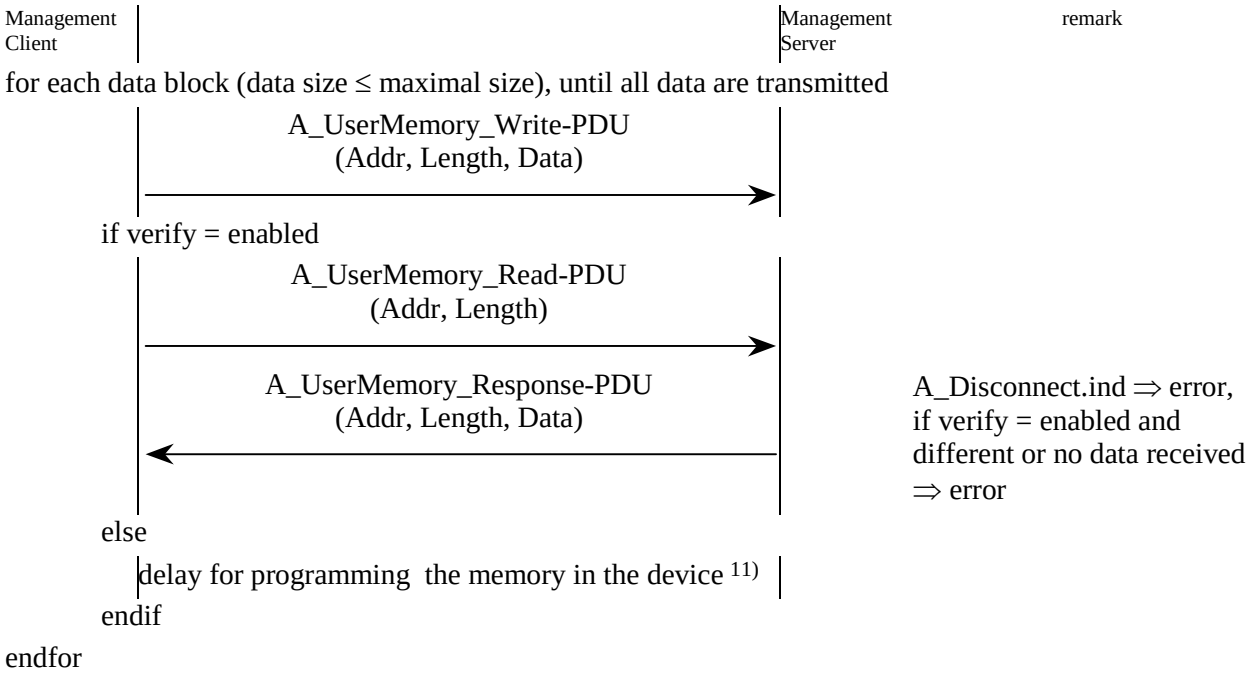
This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_Memory_Read-PDUs and/or A_Memory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 11 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_UserMemory_Write
- A_UserMemory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.19.3 Procedure: DMP_UserMemWrite_RCoV

This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall be used.

Preconditions

This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_UserMemory_Read-PDUs and/or A_UserMemory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

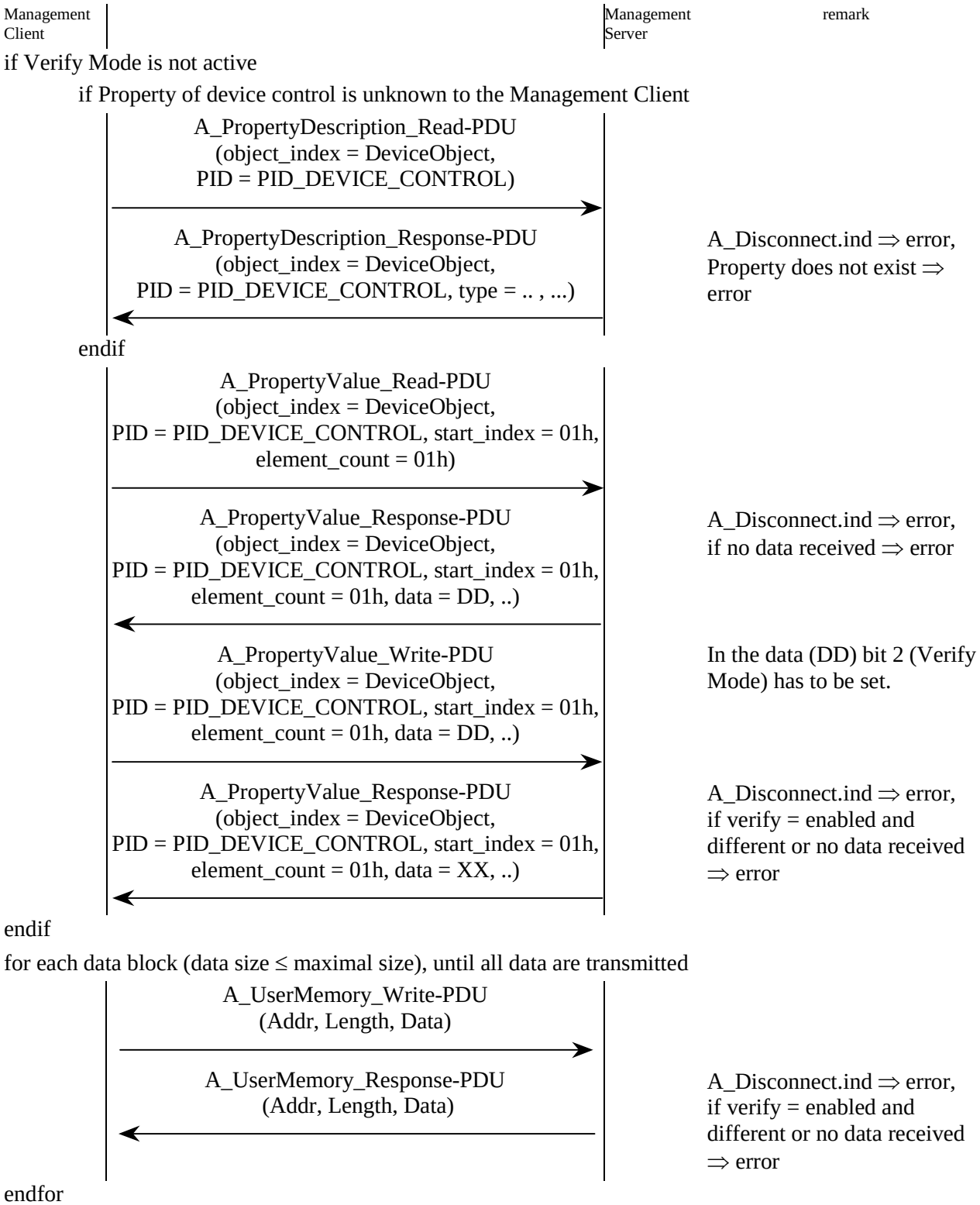
- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 11 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_UserMemory_Write

11) The delay time depends on the Management Server and on the amount of written octets (see [08]).

Sequence



Exception handling

The general exception handling shall apply.

3.20 DM_UserMemVerify

3.20.1 Use

This device Management Procedure shall read a contiguous block of user memory in the Management Server and compare it with the specified data. The data shall be located either in the management control or in the data block. Only the data that are specified in the data block shall be compared. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

DM_UserMemVerify (flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)

flags	bit 0 location of data 0: in data block 1: in management control All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress	address of first user memory octet that is compared by this Management Procedure
deviceEndAddress	address of the last user memory octet that is compared by this Management Procedure
data	the data that are compared by this Management Procedure. The data can be located in the data block or in the Management Procedure.

3.20.2 Procedure: DMP_UserMemVerify_RCo

This Management Procedure shall use the connection oriented communication mode.

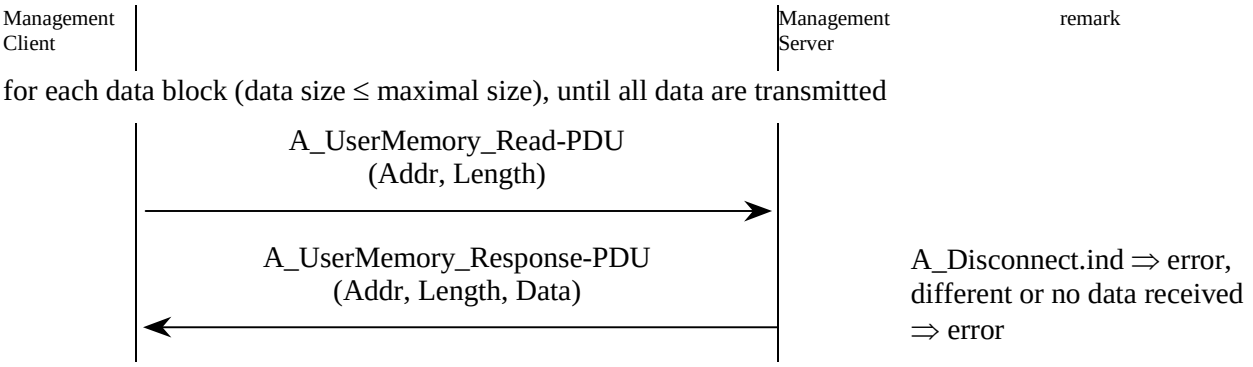
This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_UserMemory_Read-PDUs and/or A_UserMemory_Write-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 12 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.21 DM_UserMemRead

3.21.1 Use

This device Management Procedure shall read a contiguous block of memory in the Management Server and store it in the data block. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

DM_UserMemRead (flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)

flags	bit 0 location of data 0: in data block 1: - All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block.
deviceStartAddress	address of first memory octet that is read by this Management Procedure
deviceEndAddress	address of the last octet that is read by this Management Procedure
data	the data that are read by this Management Procedure. The data are stored in the data block.

3.21.2 Procedure: DMP_UserMemRead_RCo

This Management Procedure shall use the connection oriented communication mode.

Preconditions

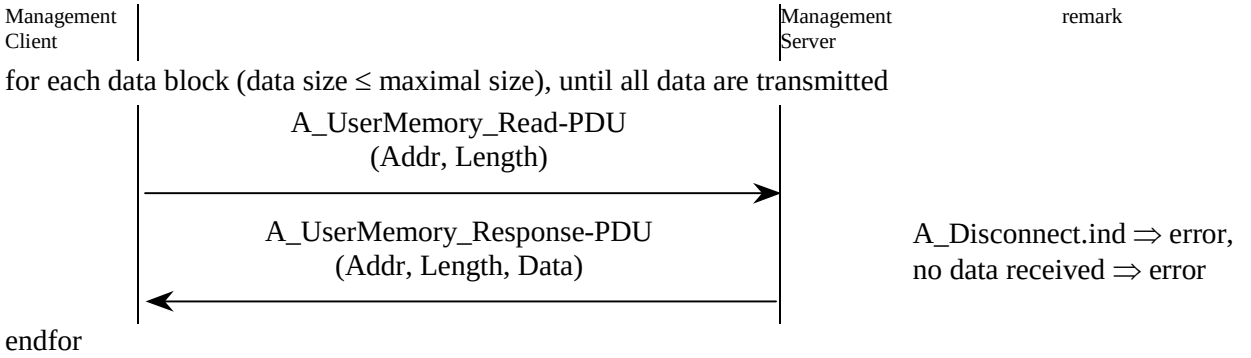
This Management Procedure shall transfer the data in datablocks and transmit these in subsequent A_Memory_Read-PDUs, as specified below, all of which except possibly the last PDU, shall have a data field (ASDU) with a size equal to the maximum size that can be transported over the communication path consisting of the Management Client, the Management Server and Couplers and Routers in between these two.

- If the Management Server does not support the L_Data_Extended frame format, then this maximal size shall be 11 octets.
- If the Management Server supports L_Data_Extended frames, then the maximal size shall be adapted in function of the capabilities of the Management Server and possible Couplers and Routers in the communication path to the Management Client. This is specified in [04].

Used Application Layer Services for Management

- A_UserMemory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.22 DM_InterfaceObjectWrite

3.22.1 Use

This device Management Procedure shall write to the specified Property value of an Interface Object. The data shall be located either in the management control or in the data block. Depending on the flag the data shall be verified immediately. The Interface Object can be addressed via the Object Type and index (e.g. first Interface Object of type ´polling master´) or via the Object Index independent of the type.

Remark

Existing implementations of Interface Object Servers in existing Management Servers may have fixed Object Indexes for the Interface Objects for Device- and Network Management. This restriction shall not be implemented in new developments. New developments of Management Clients shall instead obtain the object_index by a preceding DM_InterfaceObjectScan-procedure if the object_index is not known.

A DM_Connect shall be executed before executing this Management Procedure.

DM_InterfaceObjectWrite	(flags, dataBlockStartAddress, object_type, object_index, PID, start_index, noElements, data)
flags	bit 0: location of data 0: in data block 1: in management control bit 1: verify enabled / disabled 0: disabled 1: enabled bit 2: address mode 0: address via Object Type / index 1: address via object index All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
object_type	type of the Interface Object
object_index	index of the Interface Object of one type. This index starts counting from 0.
PID	ID of the Property
start_index	start element of the Property
noElements	number of elements that are transferred
data	the data that are transferred by this Management Procedure. The data can be located in the data block or in the Management Procedure.

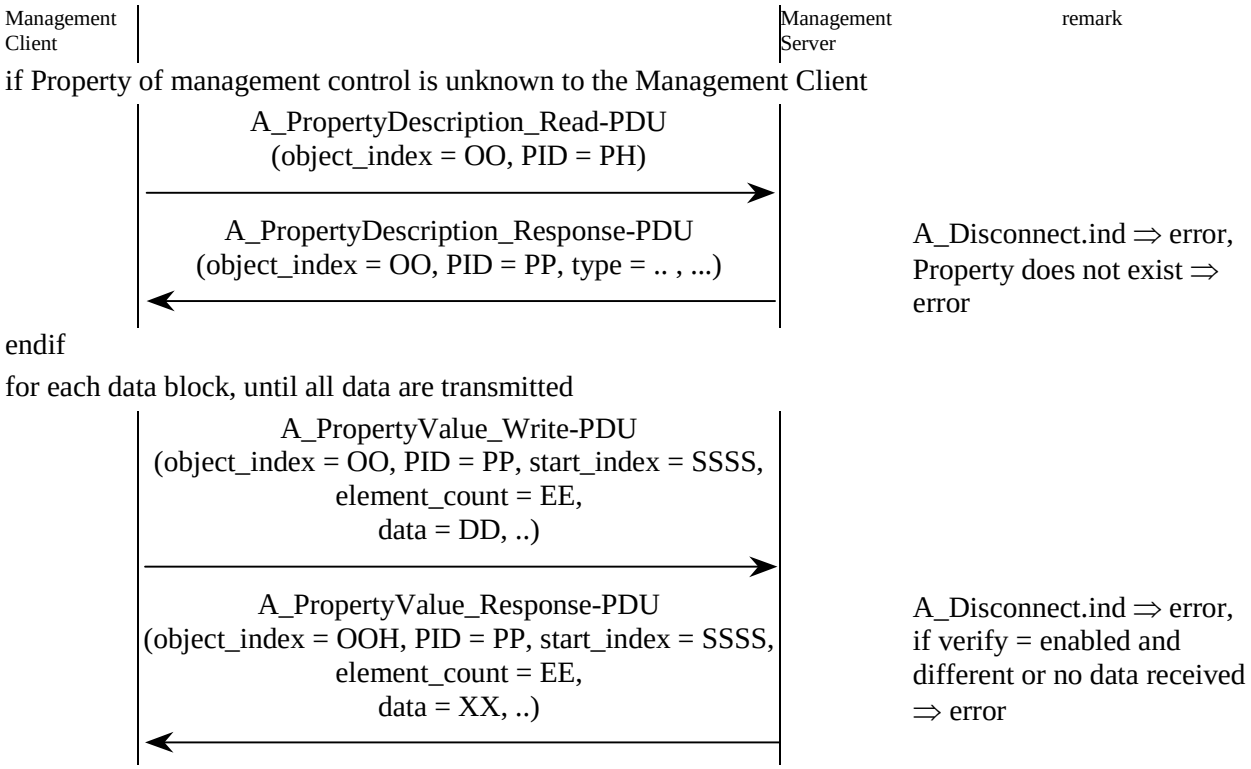
3.22.2 Procedure: DMP_InterfaceObjectWrite_R

This Management Procedure shall use the connection oriented or connectionless communication mode.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Write

Sequence



Exception handling

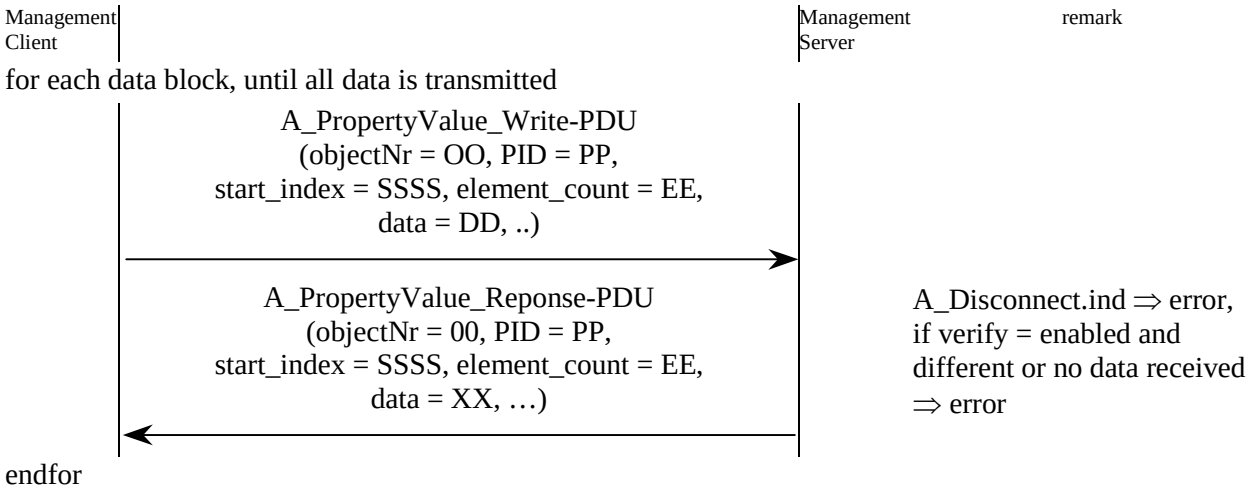
The general exception handling shall apply.

The Management Client shall not interpret the value of the Property Index contained in the A_Property-Description_Response-PDU at the level of this Management Procedure. Possibly, error handling in case an unexpected value of the Property Index can be handled at the level of the Configuration Procedure in which this Management Procedure is used.

3.22.3 Procedure: DMP_ReducedInterfaceObjectWrite_R

Prior to this procedure the procedure DMP_Connect_RCl can be executed to identify the remote device.

Sequence



3.23 DM_InterfaceObjectVerify

3.23.1 Use

This device Management Procedure shall read the Property value of an Interface Object and compare it with the specified data. The data shall be located either in the management control or in the data block. The Interface Object can be addressed via the Object Type and index (e.g. first object of type ´polling master´) or via the Object Index independent of the type.

Remark

Existing implementations of Interface Object Servers in existing Management Servers may have fixed Object Indexes for the Interface Objects for Device- and Network Management. This restriction shall not be implemented in new developments. New developments of Management Clients shall instead obtain the object_index by a preceding DM_InterfaceObject_Scan-procedure if the object_index is not known.

A DM_Connect shall be executed before executing this Management Procedure.

DM_InterfaceObjectVerify	(flags, dataBlockStartAddress, object_type, object_index, PID, start_index, noElements, data)
flags	bit 0: location of data 0: in data block 1: in management control bit 2: address mode 0: address via Object Type / index 1: address via object index All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
object_type	type of the Interface Object
object_index	index of the Interface Object of one type. This index starts counting from 0.
PID	ID of the Property
start_index	start element of the Property
noElements	number of elements, which are compared
data	the data that are compared by this Management Procedure. The data can be located in the data block or in the Management Procedure.

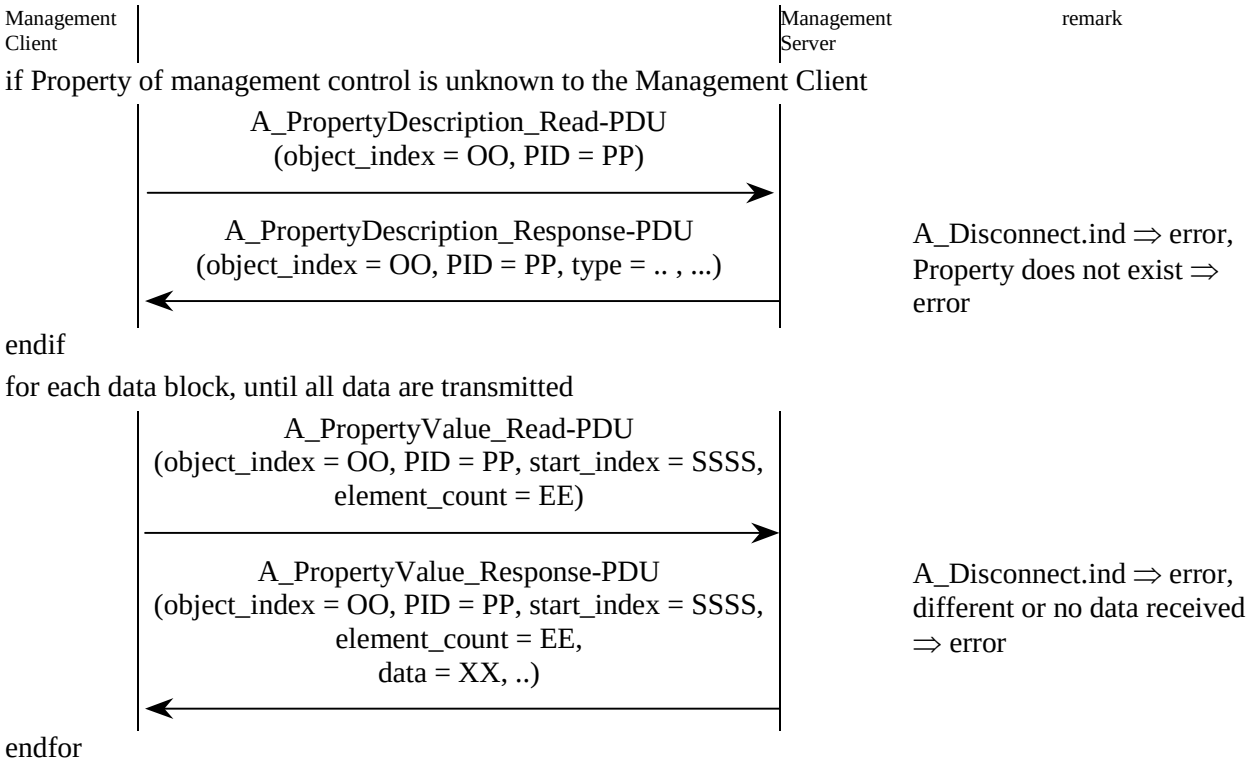
3.23.2 Procedure: DMP_InterfaceObjectVerify_R

This Management Procedure shall use the connection oriented or connectionless communication mode.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

The general exception handling shall apply.

The Management Client shall not interpret the value of the Property Index contained in the A_Property-Description_Response-PDU at the level of this Management Procedure. Possibly, error handling in case an unexpected value of the Property Index can be handled at the level of the Configuration Procedure in which this Management Procedure is used.

3.24 DM_InterfaceObjectRead

3.24.1 Use

This device Management Procedure shall read the Property value of an Interface Object and store it in the data block. The Interface Object can be addressed via the Object Type and Object Index (e.g. first Interface Object of type ´polling master´) or via the Object Index independent of the type.

Remark

Existing implementations of Interface Object Servers in existing Management Servers may have fixed Object Indexes for the Interface Objects for Device- and Network Management. This restriction shall not be implemented in new developments. New developments of Management Clients shall instead obtain the object_index by a preceding DM_InterfaceObject_Scan-procedure if the object_index is not known.

A DM_Connect shall be executed before executing this Management Procedure.

DM_InterfaceObjectRead	(flags, dataBlockStartAddress, object_type, object_index, PID, start_index, noElements, data)
flags	bit 0: location of data 0: in data block 1: - bit 2: address mode 0: address via Object Type / index 1: address via object index All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
object_type	type of the Interface Object
object_index	index of the Interface Object of one type. This index starts counting from 0.
PID	ID of the Property
start_index	start element of the Property
noElements	number of elements that are read
data	the data that are read by this Management Procedure. The data are stored in the data block.

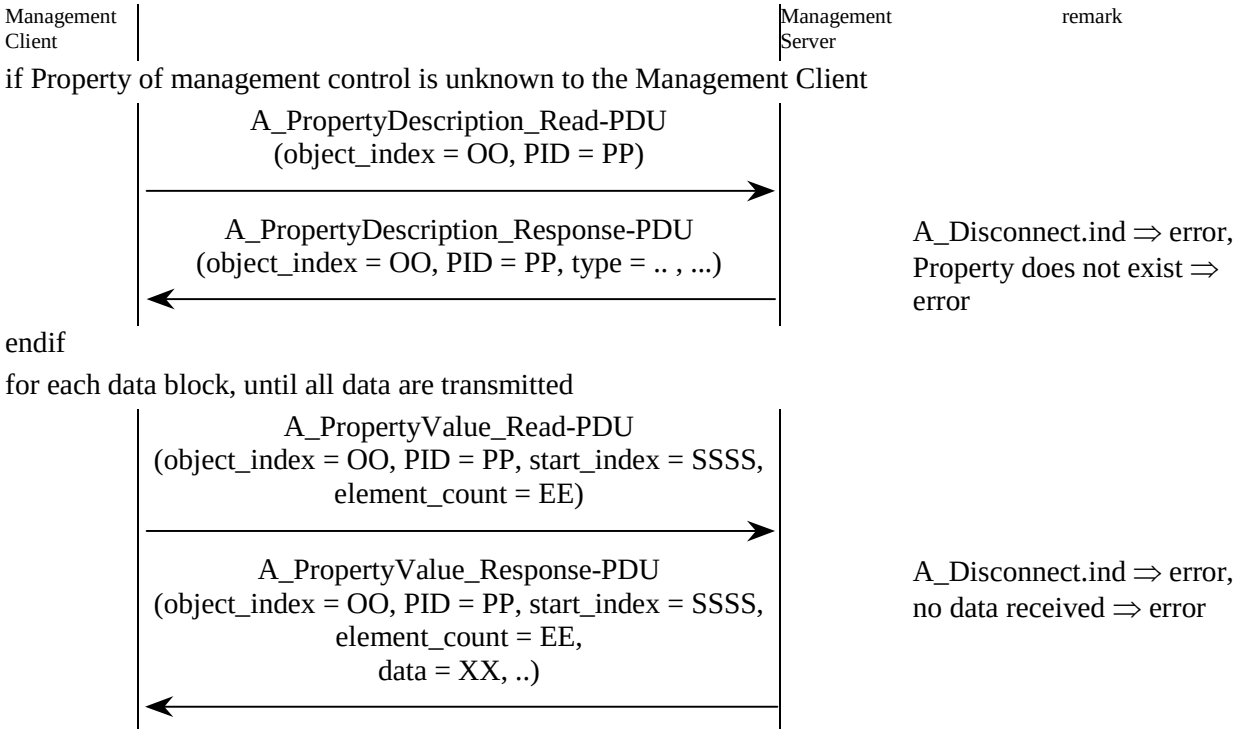
3.24.2 Procedure: DMP_InterfaceObjectRead_R

This Management Procedure shall use the connection oriented or connectionless communication mode.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

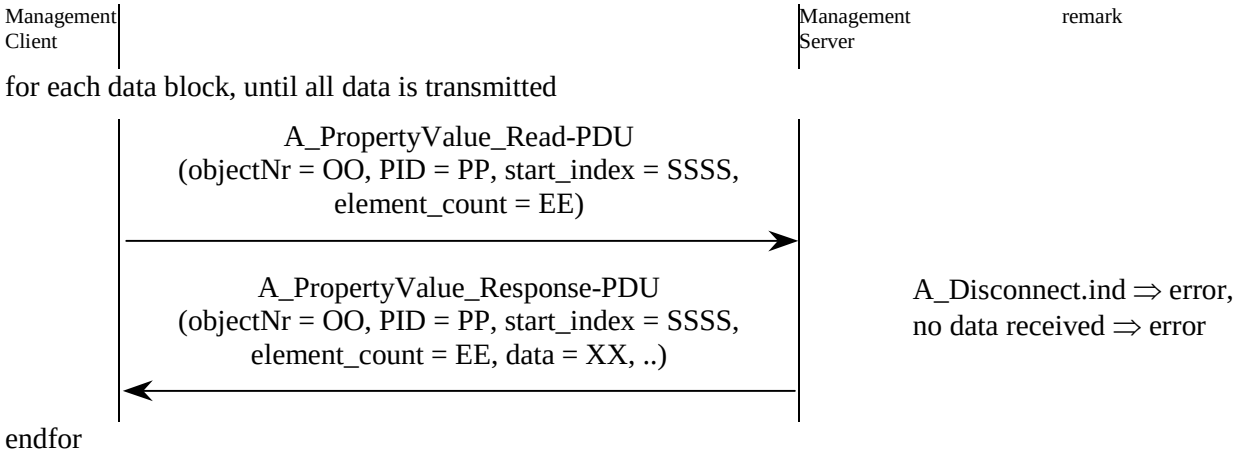
The general exception handling shall apply.

The Management Client shall not interpret the value of the Property Index contained in the A_Property-Description_Response-PDU at the level of this Management Procedure. Possibly, error handling in case an unexpected value of the Property Index can be handled at the level of the Configuration Procedure in which this Management Procedure is used.

3.24.3 Procedure: DMP_ReducedInterfaceObjectRead_R

Prior to this procedure the procedure DMP_Connect_RCl can be executed to identify the remote device.

Sequence



3.25 DM_InterfaceObjectScan

3.25.1 Use

This device Management Procedure shall scan for the available the Interface Objects in one Management Server and return the description of each found Interface Object.

A DM_Connect shall be executed before executing this Management Procedure.

DM_InterfaceObjectScan	(flags, dataBlockStartAddress, object_index, data)
flags	bit 0: location of data 0: in data block 1: no data returned bit 2: scan all properties 0: read only the type of the Interface Object 1: scan all the properties of the Interface Object bit 3: scan Interface Objects 0: read only the data of one object 1: scan all Interface Objects All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
object_index	index of the Interface Object. If Interface Object scan is enabled the value shall be 0.
data	the data that are read by this Management Procedure. The data are stored in the data block.

For each found Interface Object the following data is returned - the end of the list is marked by Interface Object nr. 0:

- object_index
- object_type
- PropertyCount

For each found Property the following data shall be returned:

- PropertyIndex
- PID
- Data Type
- Number of elements
- Access rights / write enable

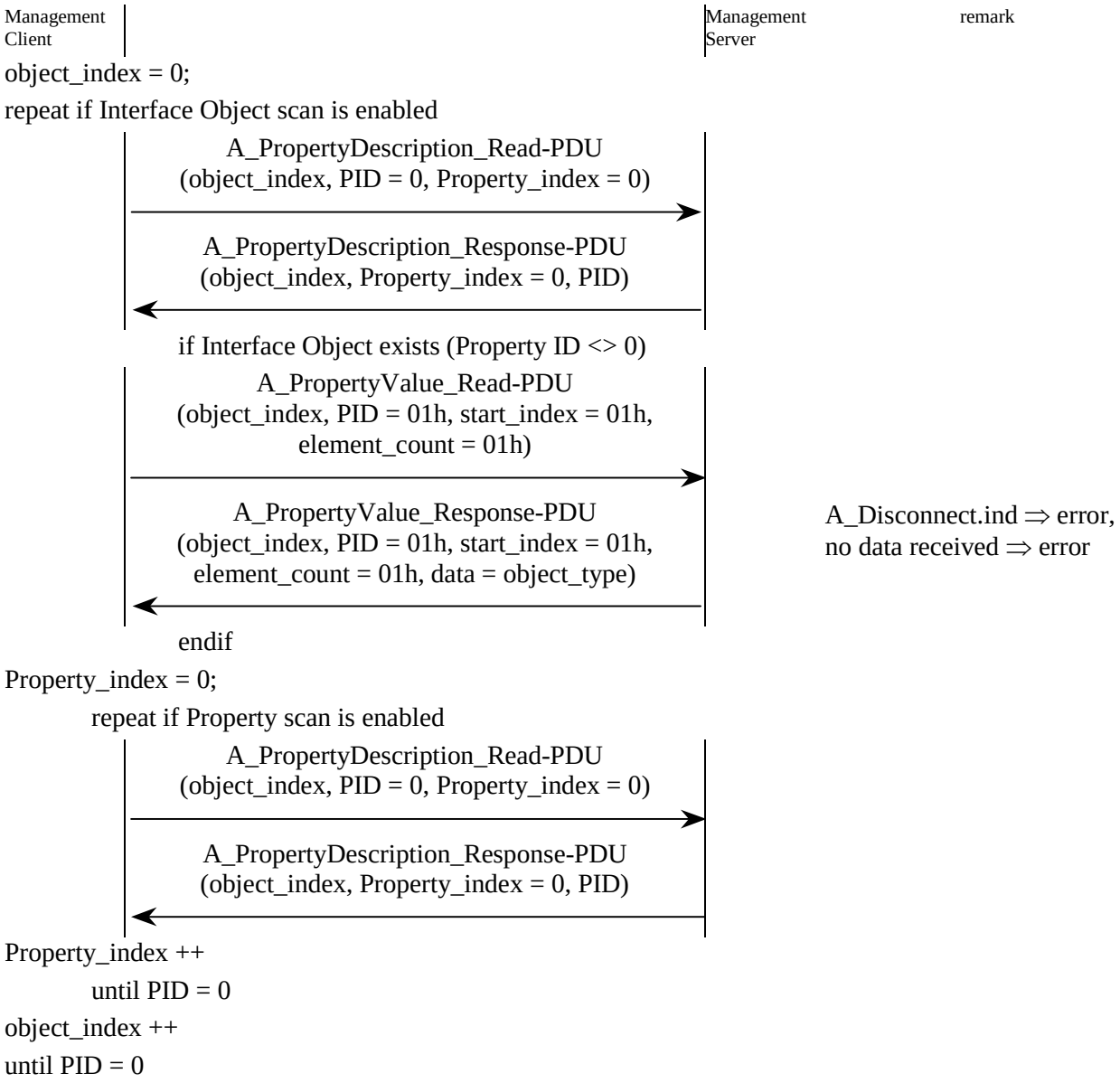
3.25.2 Procedure: DMP_InterfaceObjectScan_R

This Management Procedure shall use the connection oriented or connectionless communication mode.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence

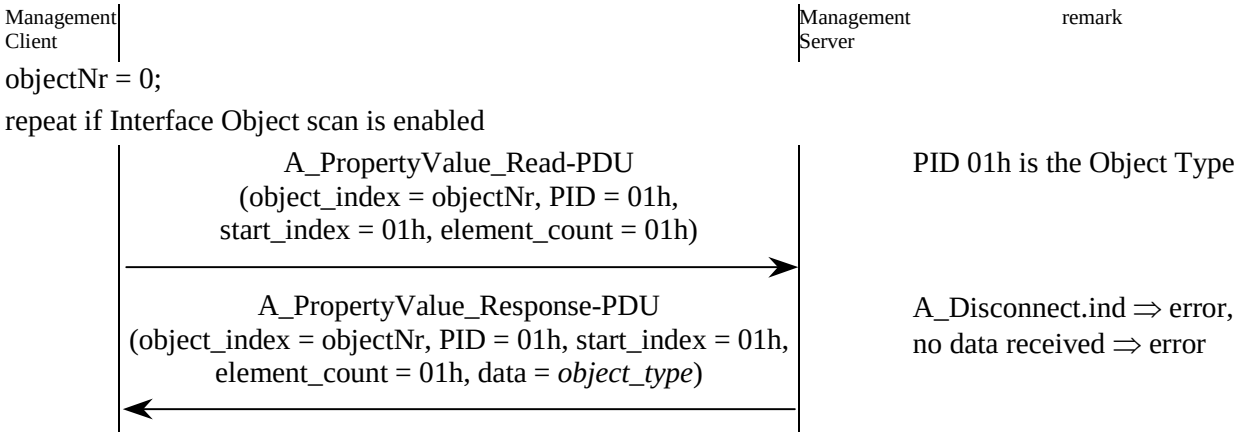

Management Client
Management Server
remark
object_index = 0;
repeat if Interface Object scan is enabled
A_PropertyDescription_Read-PDU (object_index, PID = 0, Property_index = 0)
A_PropertyDescription_Response-PDU (object_index, Property_index = 0, PID)
if Interface Object exists (Property ID <> 0)
A_PropertyValue_Read-PDU (object_index, PID = 01h, start_index = 01h, element_count = 01h)
A_PropertyValue_Response-PDU (object_index, PID = 01h, start_index = 01h, element_count = 01h, data = object_type)
A_Disconnect.ind ⇒ error, no data received ⇒ error
endif
Property_index = 0;
repeat if Property scan is enabled
A_PropertyDescription_Read-PDU (object_index, PID = 0, Property_index = 0)
A_PropertyDescription_Response-PDU (object_index, Property_index = 0, PID)
Property_index ++
until PID = 0
object_index ++
until PID = 0

3.25.3 Procedure: DMP_ReducedInterfaceObjectScan_R

Use

This Management Procedure shall scan the Interface Objects in a device with Reduced Interface Objects. Prior to this Management Procedure the Management Procedure DMP_Connect_RCl can be executed to identify the remote device.

Sequence



Exception handling

The general exception handling shall apply.

3.26 DM_InterfaceObjectInfoReport

3.26.1 Use

This Management Procedure shall be used by a Management Server to report the value of an Interface Object Property to any interested communication partner.

Used Application Layer Services for Management

- A_NetworkParameter_InfoReport

Parameters of the Management Procedure

DMP_InterfaceObjectInfoReport_RCl (/* [in] */ mpp_ASAP, /* [in] */ mpp_comm_mode,
/* [in] */ mpp_hop_count_type, /* [in] */ mpp_object_type, /* [in] */ mpp_PID,
/* [in] */ mpp_priority, /* [in] */ mpp_test_info, /* [in] */ mpp_test_result)

mpp_ASAP:	The parameter ASAP shall only be evaluated if the parameter mpp_comm_mode equals point-to-point connectionless. It shall in this case contain the Individual Address of the communication partner to which the A_NetworkParameter_InfoReport-PDU shall be sent.

mpp_comm_mode:	Communication mode that shall be used by the Management Server for transmission of the A_NetworkParameter_InfoReport-PDU. It can be – point-to-all-points connectionless (this is broadcast), or – point-to-point connectionless.
mpp_hop_count_type:	Value of the hop_count that shall be used to transmit the A_NetworkParameter_InfoReport-PDU. NOTE 7 This value shall be fixed in function of the specific Property of which the value shall be reported via this Management Procedure..
mpp_object_type:	Value that shall be used by the Management Server for the subfield object_type of the field parameter_type of the A_NetworkParameter_InfoReport-PDU.
mpp_PID:	Value that shall be used by the Management Server for the subfield PID of the field parameter_type of the A_NetworkParameter_InfoReport-PDU.
mpp_priority:	The priority that shall be used for the transmission of the A_NetworkParameter_InfoReport-PDU.
mpp_test_info:	This shall be the contents of the field tests_info of the A_NetworkParameter_InfoReport-PDU.
mpp_test_result:	This shall be the contents of the field tests_result of the A_NetworkParameter_InfoReport-PDU.

Sequence

Management Client		Management Server	remark
			This procedure shall be initiated by the Management Server. According the Management Procedure Parameter mpp_comm_mode, it shall send the A_NetworkParameter_InfoReport-PDU * on broadcast communication mode * or on point-to-point connectionless communication mode to the Individual Address given in mpp_ASAP as given below. The priority shall be as requested in mpp_priority. The hop_count_type shall be requested as in the Parameter mpp_hop_count_type.
	A_NetworkParameter_InfoReport-PDU (object_type = mpp_object_type, property_id = mpp_PID, test_info = mpp_tests_info, test_result = mpp_test_result) →		

Example use (informative)

EXAMPLE 5 The A_NetworkParameter_InfoReport is used in Flexible E-Mode Channels (see [04]) to report a human localisation action of an E-Mode Channel, using PID_LOCALISATION_REPORT ([03]) as follows:

```
/* Report on a localisation action on E-Mode Channel n. */
DMP_InterfaceObjectInfoReport_RCl(mpp_ASAP = Controller.IA, mpp_Obj.Type = E-Mode Device Object,
    mpp_Prop.ID = PID_LOCALISATION_REPORT, mpp_Test_Info = 00h, mpp_Test_Result = Channel Number)
```

3.27 DM_FunctionProperty_Write_R

3.27.1 Use

This Management Procedure shall use point-to-point connection-oriented or point-to-point connectionless communication mode. In case the point-to-point connection-oriented communication mode is used, a DMP_Connect_RCo shall be performed preceding this procedure.

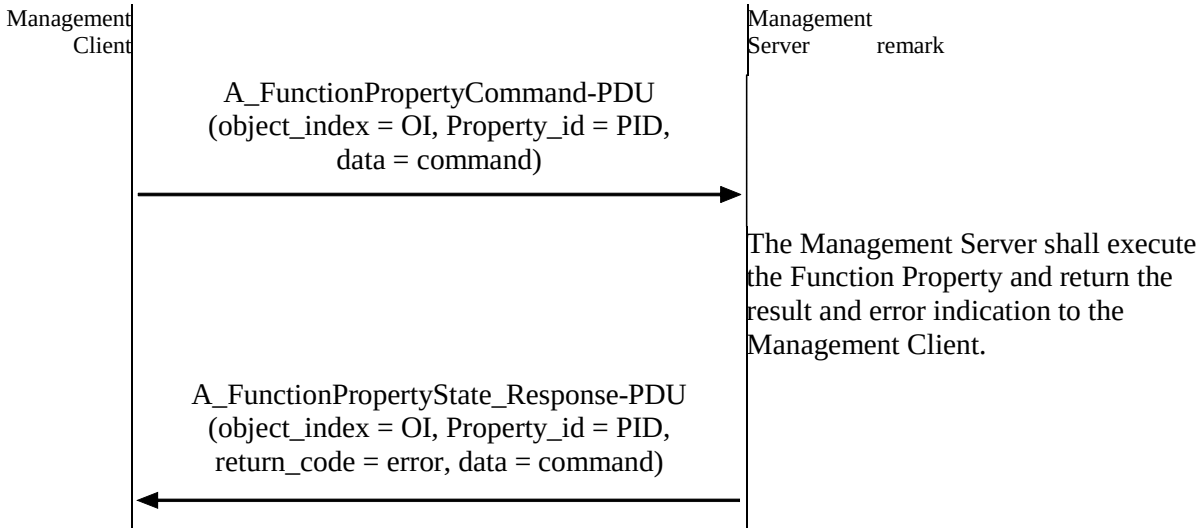
Used Application Layer Services for Management

- A_FunctionPropertyCommand

Parameters used during this Management Procedure

- OI: Object Index of the Interface Object in which the Function Property is located.
- PID: Property Identifier of the Function Property
- command: The command that is requested of the Function Property. The coding shall be Function Property specific and is specified in [03]
- error: Error code returned by the Function Property Server.

Sequence



Error handling

The error shall be Function Property specific and is specified in [03]. The handling of this error depends on the Configuration Procedure in which this Management Procedure is used.

3.28 DM_LoadStateMachineWrite

3.28.1 Use

This device Management Procedure shall be used to write to the Load State Machine of a Management Server. The data are located in the Management Procedure. Depending on the flag the resulting state shall be verified immediately.

The Load State Machines are specified in [03].

A DM_Connect shall be executed before executing this Management Procedure.

DM_LoadStateMachineWrite (flags, stateMachineType, stateMachineNr, event, eventData)

flags bit 0 : location of data
0: -
1: in management control
bit 1 : verify resulting state enabled / disabled
0: disabled
1: enabled
All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

stateMachineType type of the object that contains the state machine:

type	state machine
0001	address table
0002	association table
0003	application program
0004	PEI program

stateMachineNr index to the state machine. For this index only the state machines of this type are relevant. This index starts counting from 0.

event code of the event

eventData data of the event

Different events can be sent to the Load State Machines.

State Machine type / Event	address table	association table	application program	PEI program
Unload	X	X	X	X
Load	X	X	X	X
LoadCompleted	X	X	X	X
AllocAbsDataSeg	X	X	X	X
AllocAbsStackSeg			X	X
AllocAbsTaskSeg	X	X	X	X
TaskPtr			X	
TaskCtrl1			X	
TaskCtrl2			X	

3.28.2 Procedure: DMP_LoadStateMachineWrite_RCo_Mem

Use

This Management Procedure shall use the connection oriented communication mode.

The control and state of the Load State Machine shall be located in the memory of the Management Server and shall be accessible as Memory Mapped Resource.

The Verify Mode of the Management Server shall not be used.

This Management Procedure shall support only one state machine of each type.

This Management Procedure shall only be used with device model for mask version 070nh (BIM M112). The address of the management control is 0104h. The address (AAAA) of the load state depends on the Load State Machine.

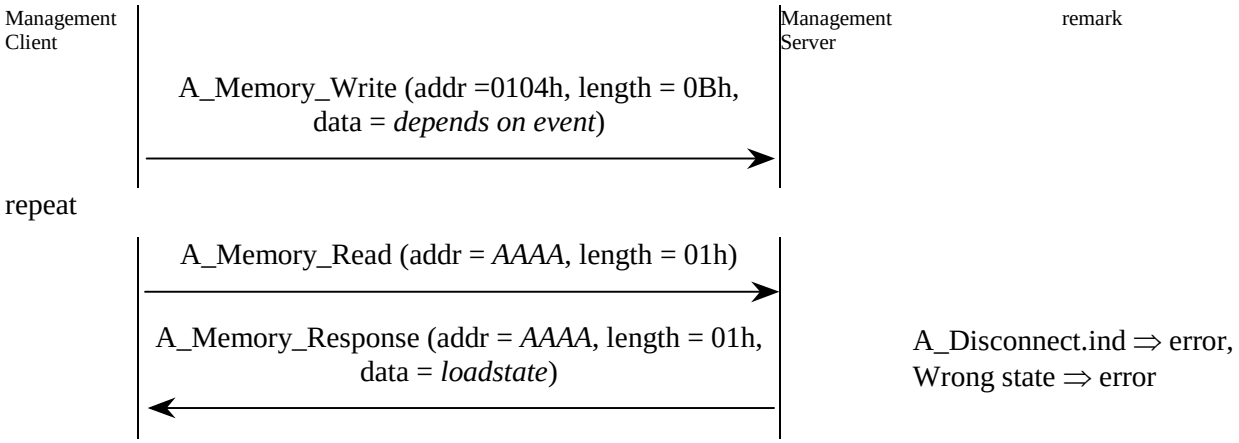
state machine	address of load state (*AAAA*)
address table	B6EAh
association table	B6EBh
application program	B6ECh
PEI program	B6EDh

This Management Procedure shall not be used for further developments of Management Servers.

Used Application Layer Services for Management

- A_Memory_Write
- A_Memory_Read

Sequence



The transmitted data depend on the event.

- LoadEvent: Unload

data	
state machine / event	reserved
*L*4	00h
1 octet	10 octets

- LoadEvent: Load

data	
state machine / event	reserved
*L*1	00h
1 octet	10 octets

- LoadEvent: LoadComplete

data	
state machine / event	reserved
*L*2	00h
1 octet	10 octets

- LoadEvent: AllocAbsDataSeg (segment type 0)

data								
state machine / event	segment type	segment ID	start address	length	access attributes	memory type	memory attributes	reserved
*L*3	00h	00h	*SSSS*	*EEEE - SSSS* +1	*AA*	*TT*	*MM*	00h
1 octet	1 octet	1 octet	2 octets	2 octets	1 octet	1 octet	1 octet	1 octet

Access Attributes — contains the access level of the segment
- bit 0…3 write access level
- bit 4…7 read access level

Memory type — contains the type of the memory of the segment
- bit 0…2 memory type
  - 1 Zero page RAM
  - 2 RAM
  - 3 EEPROM
- bit 3…7 Reserved. Shall be zero

Memory Attributes — additional memory configuration
- bit 0…6 Reserved. Shall be zero
- bit 7
  - 0 Checksum control disabled
  - 1 Checksum control enabled

- LoadEvent: AllocAbsStackSeg (segment type 1)

data								
state machine / event	segment type	segment ID	start address	length	access attributes	memory type	memory attributes	reserved
*L*3	01h	00h	*SSSS*	*EEEE - SSSS +1*	*AA*	*TT*	*MM*	00h
1 octet	1 octet	1 octet	2 octets	2 octets	1 octet	1 octet	1 octet	1 octet

Access Attributes	contains the access level of the segment bit 0…3 write access level bit 4…7 read access level
Memory type	contains the type of the memory of the segment bit 0…2 memory type 1 Zero page RAM 2 RAM 3 EEPROM bit 3…7 Reserved. Shall be zero
Memory Attributes	additional memory configuration bit 0…6 Reserved. Shall be zero bit 7 0 Checksum control disabled 1 Checksum control enabled

- LoadEvent: AllocAbsTaskSeg (segment type 2)

data					
state machine / event	segment type	segment ID	start address	PEI type	application ID / table ID - Version
*L*3	02h	00h	*SSSS*	*PP*	*MM MM TT TT VV*
1 octet	1 octet	1 octet	2 octets	1 octet	5 octets

Application ID / Table ID	Application Software Type MM MM Software Manufacturer ID TT TT Manufacturer Specific Application Software ID VV Version of the Application Software

- LoadEvent: TaskPtr (segment type 3)

data						
state machine/ event	segment type	segment ID	initAddr	SaveAddr	PEIhandler	reserved
*L*3	03h	00h	*IIII*	*SSSS*	*PPPP*	00h
1 octet	1 octet	1 octet	2 octets	2octets	2 octets	2 octets

- LoadEvent: TaskCtrl1 (segment type 4)

data					
state machine/ event	segment type	segment ID	Interface Object address	Interface Object count	reserved
*L*3	04h	00h	*AAAA*	*NN*	00h
1 octet	1 octet	1 octet	2 octets	1 octet	5 octets

- LoadEvent: TaskCtrl2 (segment type 5)

data						
state machine/ event	segment type	segment ID	callbackAddr	CommObjPtr	CommObjSegPtr1	CommObjSeg-Ptr2
*L*3	05h	00h	*CCCC*	*OOOO*	*1111h*	*2222h*
1 octet	1 octet	1 octet	2 octet	2octet	2 octet	2 octet

Exception handling

The general exception handling shall apply.

3.28.3 Procedure: DMP_LoadStateMachineWrite_RCo_IO

This Management Procedure shall use the connection oriented communication mode.

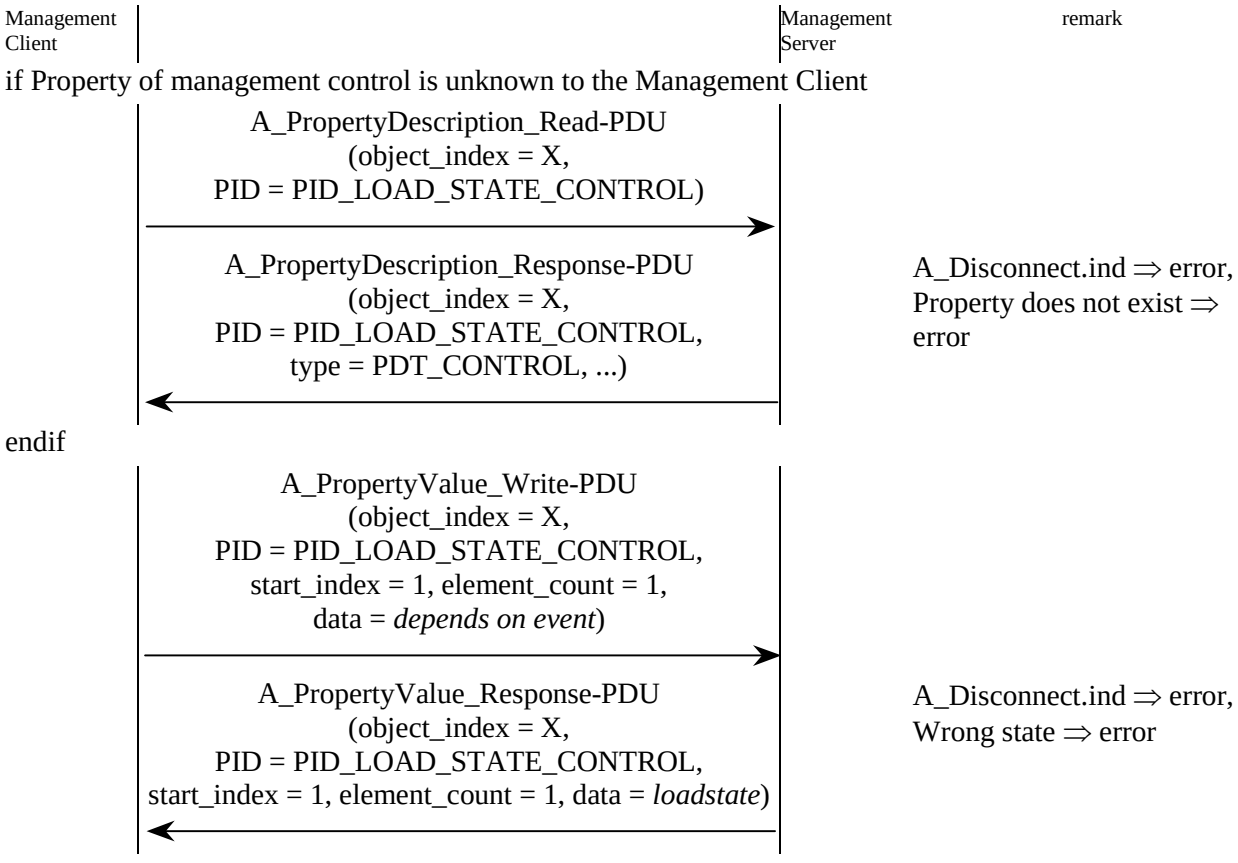
The control and state of the Load State Machine shall be located in Interface Objects of the Management Server and shall be accessible via Property services.

The Management Client shall search the according Interface Object in the Management Server.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Write

Sequence



The transmitted data depend on the event.

3.28.3.1 LoadEvent: Unload (write)

data	
event	reserved
04h	00h
1 octet	9 octets

This command shall unload a loadable part. All data shall be declared as invalid. The Load State Machine shall change to Unloaded.

3.28.3.2 LoadEvent: Start Loading (write)

data	
event	reserved
01h	00h
1 octet	9 octets

This command shall start the loading of the loadable part. The Load State Machine shall change to Loading.

3.28.3.3 LoadEvent: LoadCompleted (write)

data	
event	reserved
02h	00h
1 octet	9 octets

This command shall complete the loading of the loadable part. The checksum shall be calculated; all data shall be declared as valid and the Load State Machine shall change to Loaded. In case of download of an executable part, this executable part shall be started if the other run conditions are fulfilled.

3.28.3.4 Load Control: Additional Load Controls (Write)

Type = Additional	Subtype	Data	If data less than 8 octets: Fill octets
03h	xx	Data depending on the sub type	.. 00h 00h 00h 00h 00h 00h 00h
1 octet	1 octet	8 octets	

Please refer to [08] for the requirements on which subtype shall be supported per Profile.

- LoadEvent: AllocAbsDataSeg (segment type 0)

data							
event	segment type	start address	length	access attributes	memory type	memory attributes	reserved
03h	00h	*SSSS*	*EEEE* - *SSSS* +1	*AA*	*TT*	*MM*	00h
1 octet	1 octet	2 octet	2 octet	1 octet	1 octet	1 octet	1 octet

This load event shall serve for the absolute allocation of data or code.

Access Attributes contains the access level of the segment
- bit 0…3 write access level
- bit 4…7 read access level

Memory type contains the type of the memory of the segment
- bit 0…2 memory type
  - 1 Zero page RAM
  - 2 RAM
  - 3 EEPROM
- bit 3…7 Reserved. Shall be zero

Memory Attributes additional memory configuration
- bit 0…6 Reserved. Shall be zero
- bit 7 0 Checksum control disabled

- LoadEvent: AllocAbsStackSeg (segment type 1)

data							
event	segment type	start address	length	access attributes	memory type	memory attributes	reserved
03h	01h	*SSSS*	*EEEE* - *SSSS* +1	*AA*	*TT*	*MM*	00h
1 octet	1 octet	2 octets	2 octets	1 octet	1 octet	1 octet	1 octet

Access Attributes contains the access level of the segment
- bit 0…3 write access level
- bit 4…7 read access level

Memory type contains the type of the memory of the segment
- bit 0…2 memory type
  - 1 Zero page RAM
  - 2 RAM
  - 3 EEPROM
- bit 3…7 Reserved. Shall be zero

Memory Attributes additional memory configuration
- bit 0…6 Reserved. Shall be zero
- bit 7 0 Checksum control disabled

- LoadEvent: AllocAbsTaskSeg (segment type 2)

data				
event	segment type	start address	PEI type	application ID / table ID - Version
03h	02h	*SSSS*	*PP*	*MM MM TT TT VV*
1 octet	1 octet	2 octets	1 octet	5 octets

Application ID / Table ID Application Software Type
- MM MM Software Manufacturer ID
- TT TT Manufacturer Specific Application Software ID
- VV Version of the Application Software

- LoadEvent: TaskPtr (segment type 3)

data					
event	segment type	initAddr	SaveAddr	PEIhandler	reserved
03h	03h	IIII	SSSS	PPPP	00h
1 octet	1 octet	2 octets	2 octets	2 octets	2 octets

- LoadEvent: TaskCtrl1 (segment type 4)

data				
event	segment type	Interface Object address	nr. of Interface Objects	reserved
03h	04h	*AAAA*	*NN*	00h
1 octet	1 octet	2 octets	1 octet	5 octets

- LoadEvent: TaskCtrl2 (segment type 5)

data					
event	segment type	callbackAddr	CommObjPtr	CommObjSegPtr1	CommObjSegPtr 2
03h	05h	CCCC	OOOO	1111h	2222h
1 octet	1 octet	2 octets	2 octets	2 octets	2 octets

- Load Event Relative Allocation

data

event	subtype	data	fill octets					
03h	0Ah	number of octets	00h	00h	00h	00h	00h	00h
8 bit	1 octet	2 octets	6 octets					

This command shall set the maximum size (in octets) of the loadable part being loaded. If the requested number of octets is not supported by the Management Server (device) then the Load State Machine of the loadable part shall change to error.

EXAMPLE This command may be used to set the maximum length of the Group Address Table or Group Association Table being downloaded.

- Load Event Data Relative Allocation

data

event	subtype	data	mode	fill	reserved	
03h	0Bh	requested memory size	00h	00h	00h	00h
8 bit	1 octet	4 octets	1 octet	1 octet	2 octets	

Mode

- bit 0 (lsb) 0: keep the existing memory contents of the allocated memory unchanged
  1: fill the memory contents of the allocated memory with the value as specified in the field *fill*.
- bit 1 to bit 7 (msb) These bits are reserved and shall be 0.

fill memory fill byte
This is the value with which the allocated memory shall be filled if bit 0 of *Mode* is set.

3.28.3.5 Load State (Response Value from Load State Property)

State
8 bit

This state value shall represent the current state of the Load State Machine.

3.28.3.6 Load Control: No Operation (Write)

00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
8 bit	9 octets								

This command shall have no effect.

3.28.3.7 Exception handling

The general exception handling shall apply.

3.28.4 Procedure: DMP_DownloadLoadablePart_RCo_IO

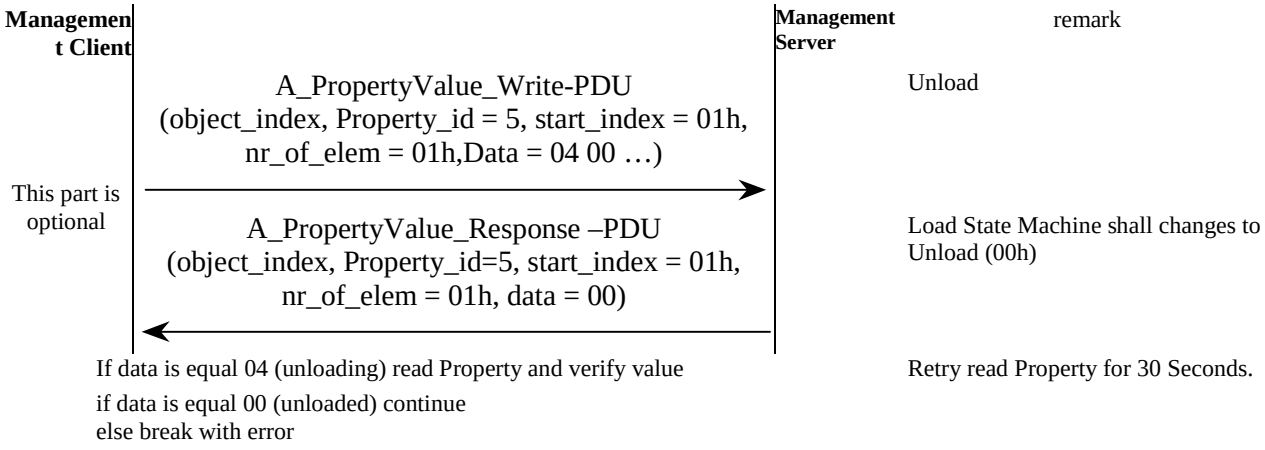
3.28.4.1 Normal conditions

This method shall use the connection oriented or connectionless remote communication.

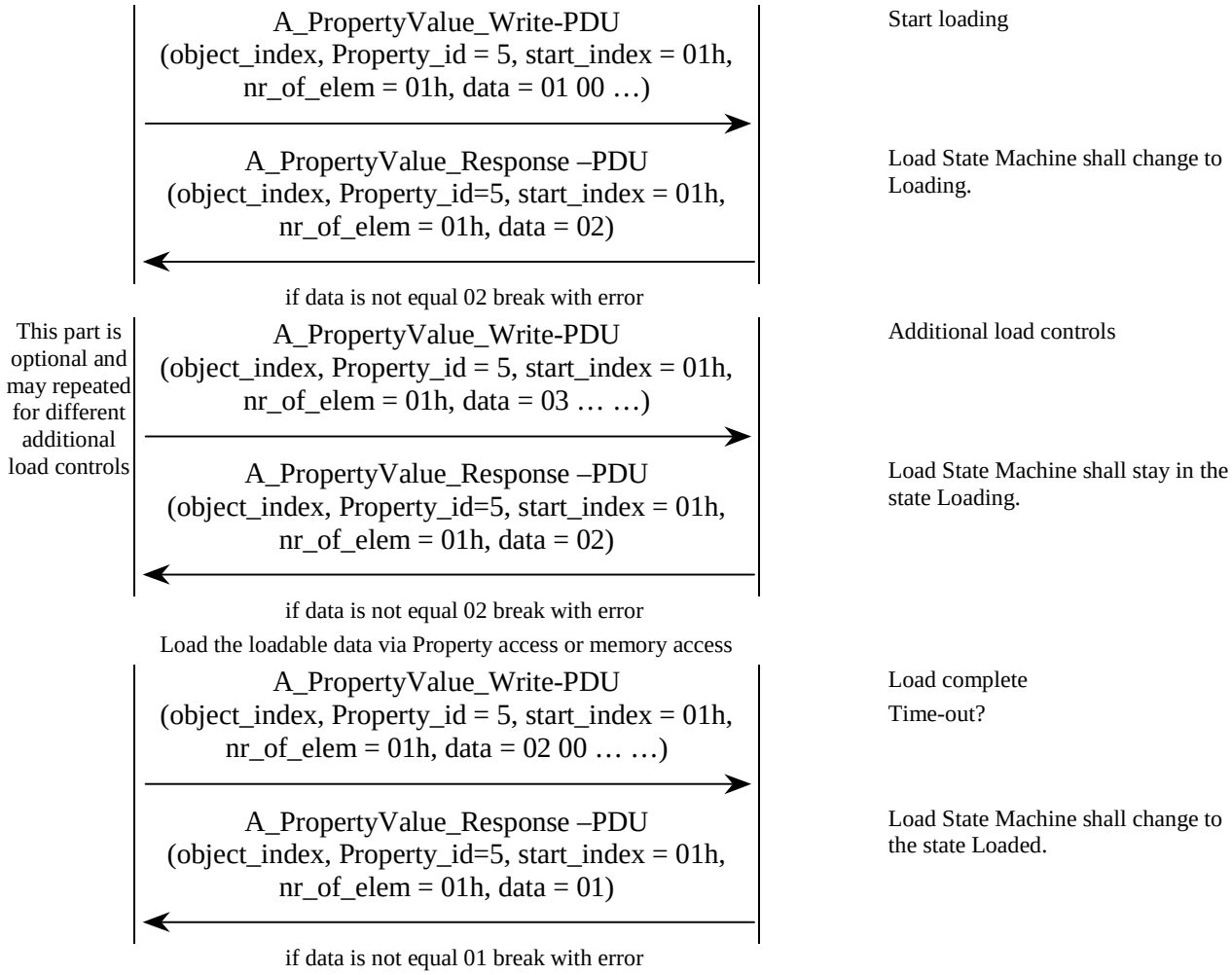
Used Application Layer Services for Management

- A_PropertyValue_Write
- A_PropertyValue_Read

Complete Download Sequence for one loadable part [12)]



12) The optional state Unloading is not used.



3.28.4.2 Error and exception handling

The general exception handling shall apply.

3.29 DM_LoadStateMachineVerify

3.29.1 Use

This device Management Procedure shall be used to verify the state of a Load State Machine of a Management Server. The state shall be located in the Management Procedure.

A DM_Connect shall be executed before executing this Management Procedure.

DM_LoadStateMachineVerify (flags, stateMachineType, stateMachineNr, state)

flags	bit 0 : location of data 0: - 1: in management control All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
stateMachineType	type of the object that contains the state machine:

type	state machine
0001	address table
0002	association table
0003	application program
0004	PEI program

stateMachineNr	index to the state machine. For this index only the state machines of this type are relevant. This index starts counting from 0.
state	state of the state machine

3.29.2 Procedure: DM_LoadStateMachineVerify_RCo_Mem

This Management Procedure shall use the connection oriented communication mode.

The control and state of the Load State Machine shall be located in the memory of the Management Server and shall be accessible as Memory Mapped Resource.

This Management Procedure shall support only one state machine of each type.

This Management Procedure shall only be used with device model for mask version 070nh (BIM M112). The address of the management control shall be 0104h. The address (*AAAA*) of the load state depends on the Load State Machine.

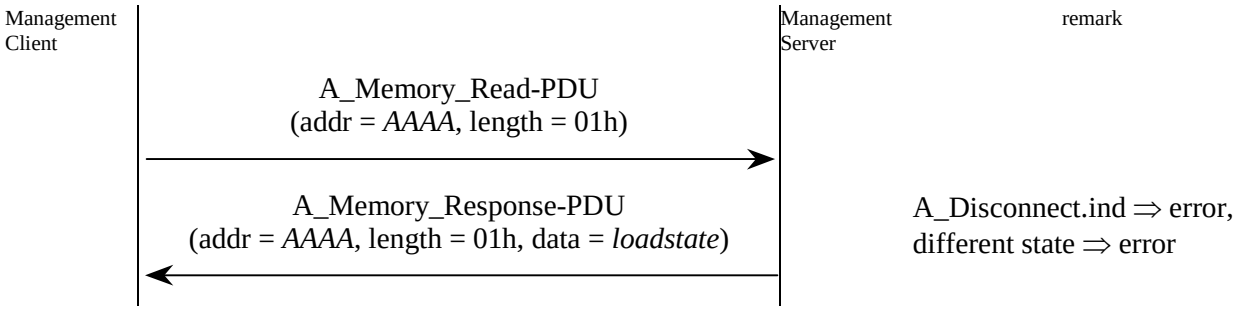
state machine	address of load state (*AAAA*)
address table	B6EAh
association table	B6EBh
application program	B6ECh
PEI program	B6EDh

This Management Procedure shall not be used for further developments of Management Servers.

Used Application Layer Services for Management

- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.29.3 Procedure: DM_LoadStateMachineVerify_R_IO

This method shall either use the connection oriented or connectionless communication mode.

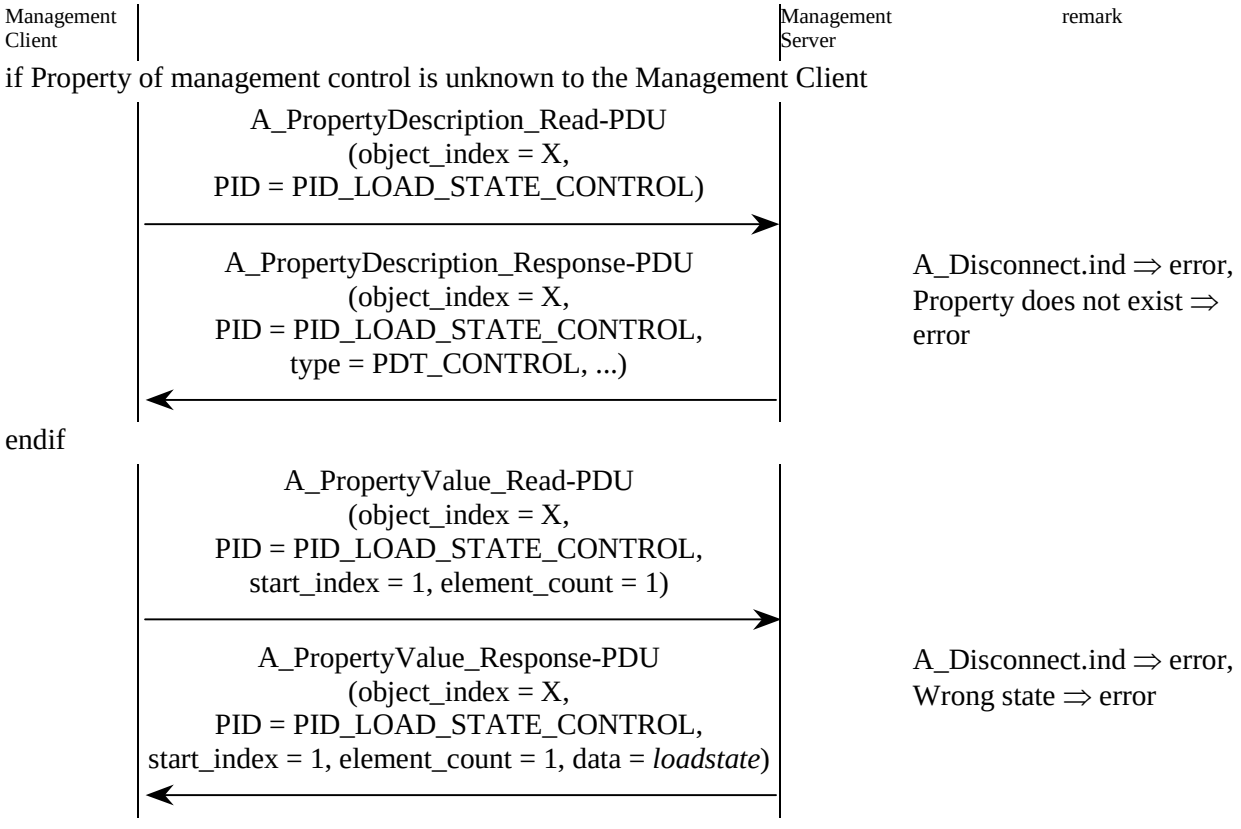
The control and state of the Load State Machine shall be located in Interface Objects of the Management Server and shall be accessible via Property services.

The Management Client shall search the according Interface Object in the Management Server.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

The general exception handling shall apply.

3.30 DM_LoadStateMachineRead

3.30.1 Use

This device Management Procedure shall be used to read the state of a Load State Machine of a Management Server. The state shall be located in the Management Procedure.

A DM_Connect shall be executed before executing this Management Procedure.

DM_LoadStateMachineRead (dataBlockStartAddress, flags, stateMachineType, stateMachineNr, state)

dataBlockStartAddress : specifies the address where the data are stored in the data block

flags : bit 0 : location of data
0: in data block
1: -
All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

stateMachineType : type of the object that contains the state machine:

type	state machine
0001	address table
0002	association table
0003	application program
0004	PEI program

stateMachineNr : index to the state machine. For this index only the state machines of this type are relevant. This index starts counting from 0.

state : state of the state machine

3.30.2 Procedure: DMP_LoadStateMachineRead_RCo_Mem

This Management Procedure shall use the connection oriented communication mode.

The control and state of the Load State Machine shall be located in the memory of the Management Server and shall be accessible as Memory Mapped Resource.

This Management Procedure shall support only one state machine of each type.

This Management Procedure shall only be used with device model for mask version 070nh (BIM M112). The address of the management control is 0104_h. The address (*AAAA*) of the load state depends on the Load State Machine.

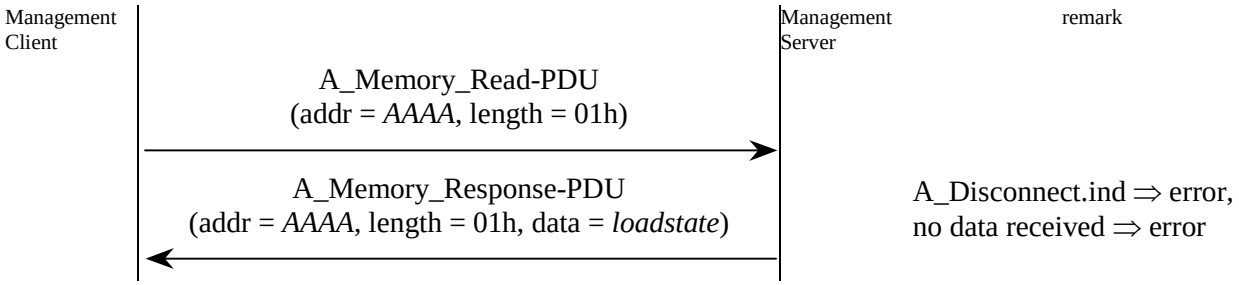
state machine	address of load state (*AAAA*)
address table	B6EAh
association table	B6EBh
application program	B6ECh
PEI program	B6EDh

This Management Procedure shall not be used for further developments of Management Servers.

Used Application Layer Services for Management

- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.30.3 Procedure: DMP_LoadStateMachineRead_R_IO

This method shall use either the connection oriented or the connectionless communication mode.

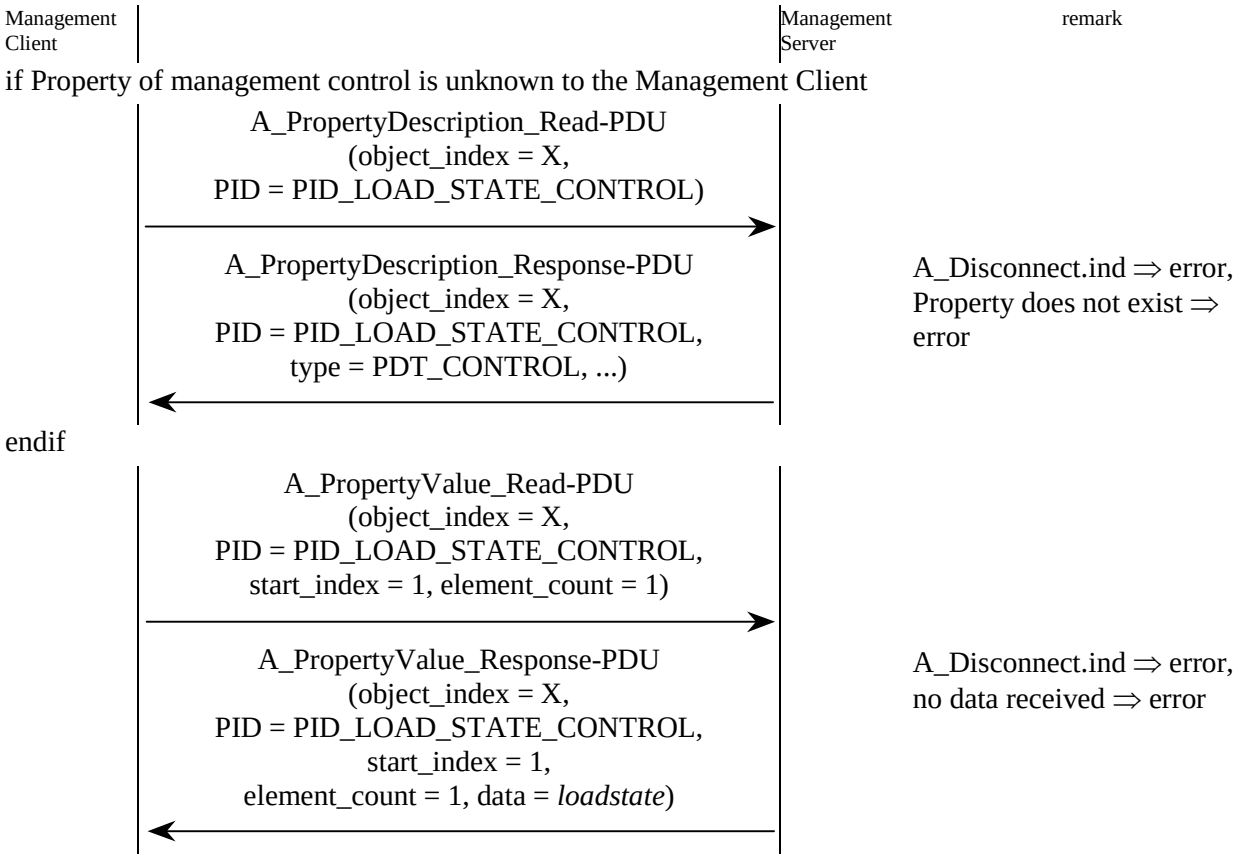
The control and state of the Load State Machine shall be located in Interface Objects of the Management Server and shall be accessible via Property services.

The Management Client shall search the according Interface Object in the Management Server.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

The general exception handling shall apply.

3.31 DM_RunStateMachineWrite

3.31.1 Use

This device Management Procedure shall be used to write to the Run State Machine of a Management Server The data shall be located in the Management Procedure. Depending on the flag, the resulting state is verified immediately.

event	resulting state
Restart	Ready or Running
Stop	Terminated

A DM_Connect shall be executed before executing this Management Procedure.

DM_RunStateMachineWrite (flags, stateMachineType, stateMachineNr, event)

flags : bit 0 : location of data
0: -
1: in management control
bit 1 : verify the resulting state enabled / disabled
0: disabled
1: enabled
All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

stateMachineType : type of the object that contains the state machine:

type	state machine
0003	application program
0004	PEI program

stateMachineNr : index to the state machine. For this index only the state machines of this type are relevant. This index starts counting from 0.

event : code of the event

3.31.2 Procedure: DMP_RunStateMachineWrite_RCo_Mem

This Management Procedure shall use the connection oriented communication mode.

The control and state of the Run State Machine shall be located in the memory of the Management Server and shall be accessible as Memory Mapped Resource.

The Verify Mode of the Management Server shall not be used.

This Management Procedure shall support only one state machine of each type.

This Management Procedure shall only be used with device model for mask version 070nh (BIM M112). The address of the run control shall be 0103h. The address (*AAAA*) of the run state depends on the Run State Machine.

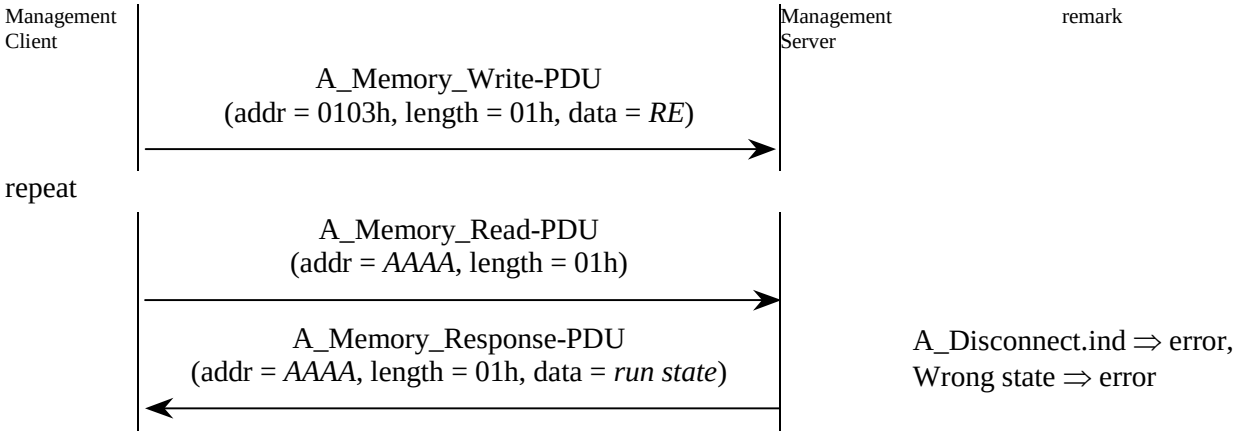
state machine	address of run state (*AAAA*)
application program	0101h
PEI program	0102h

This Management Procedure shall not be used for further developments of Management Servers.

Used Application Layer Services for Management

- A_Memory_Write
- A_Memory_Read

Sequence



Exception handling

The general exception handling shall apply.

3.31.3 Procedure: DMP_RunStateMachineWrite_R_IO

This method shall use either the connection oriented or the connectionless communication mode.

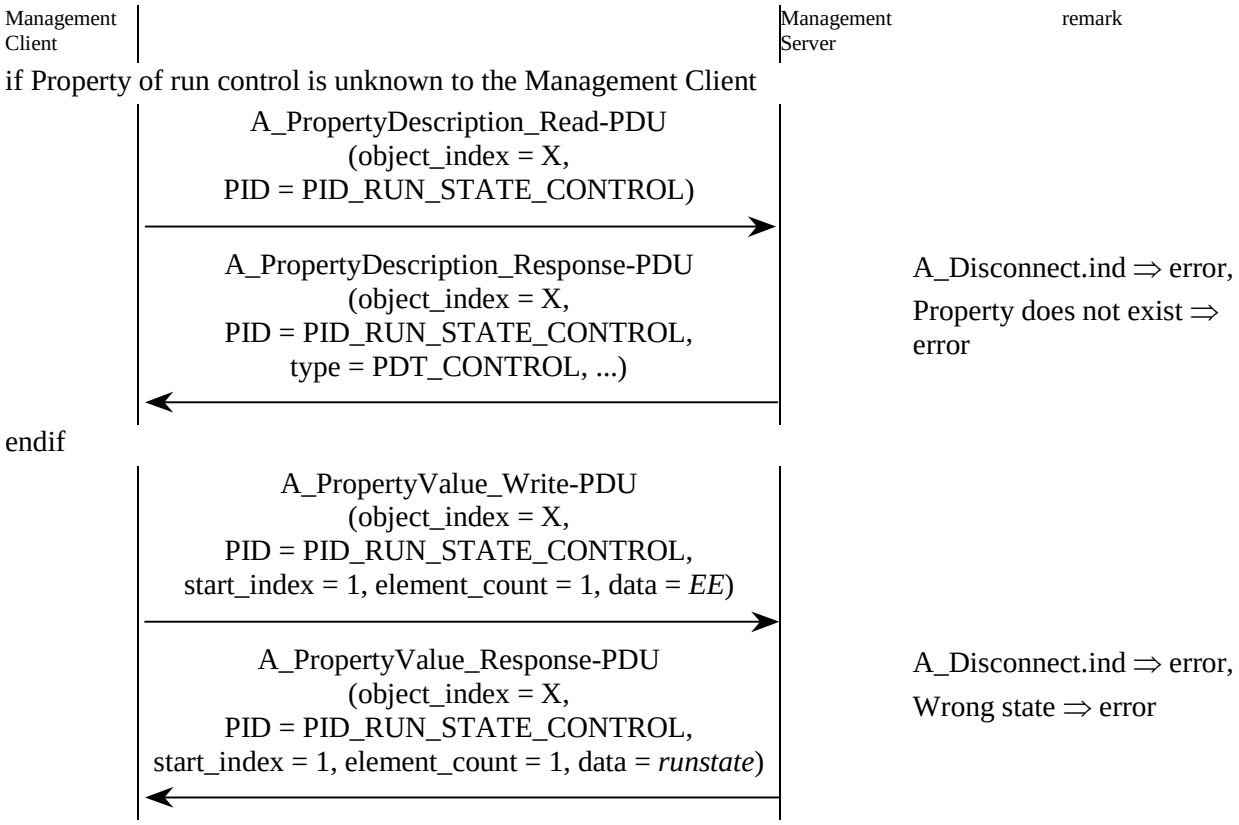
The control and state of the Run State Machine shall be located in Interface Objects of the Management Server and shall be accessible via Property services.

The Management Client shall search the according Interface Object in the Management Server.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Write

Sequence



Exception handling

The transmitted data depend on the event and are specified in the clauses 3.31.3.1 to 3.31.3.4 below.

3.31.3.1 Run Control: Restart (Write)

01h	00h	00h	00h	00h	00h	00h	00h	00h	00h
8 bit					9 octets				

This command shall restart the executable part related to the Interface Object in which this Run Control Property is located.

3.31.3.2 Run Control: Stop (Write)

02h	00h	00h	00h	00h	00h	00h	00h	00h	00h
8 bit					9 octets				

This command shall stop the executable part related to the Interface Object in which this Run Control Property is located.

3.31.3.3 Run Control: No Operation (Write)

00h	00h	00h	00h	00h	00h	00h	00h	00h	00h
8 bit					9 octets				

This command shall have no effect.

3.31.3.4 Run State (Read from Run Control)

State
8 bit

This state value shall be the Run State of the Run State Machine.

3.32 DM_RunStateMachineVerify

3.32.1 Use

This device Management Procedure shall be used to verify the state of a Run State Machine of a Management Server. The state shall be located in the Management Procedure.

A DM_Connect shall be executed before executing this Management Procedure.

DM_RunStateMachineVerify (flags, stateMachineType, stateMachineNr, state)

flags
bit 0: location of data
0: -
1: in management control
All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.

stateMachineType
type of the object that contains the state machine

state machine	type
application program	0003
PEI program	0004

stateMachineNr
index to the state machine. For this index only the state machines of this type are relevant. This index starts counting from 0.

state
state of the state machine

3.32.2 Procedure: DMP_RunStateMachineVerify_RCo_Mem

This Management Procedure shall use the connection oriented communication mode.

The control and state of the Run State Machine shall be located in the memory of the Management Server and shall be accessible as Memory Mapped Resource.

This Management Procedure shall support only one state machine of each type.

This Management Procedure shall only be used with device model for mask version 070nh (BIM M112). The address (*AAAA*) of the run state depends on the Run State Machine.

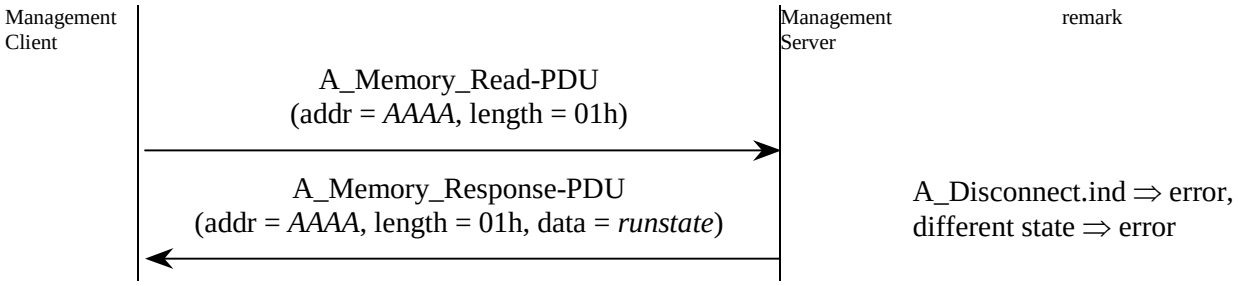
state machine	address of run state
application program	0101h
PEI program	0102h

This Management Procedure shall not be used for further developments of Management Servers.

Used Application Layer Services for Management

- A_Memory_Read

Sequence



3.32.3 Procedure: DMP_RunStateMachineVerify_R_IO

This method shall use either the connection oriented or the connectionless communication mode.

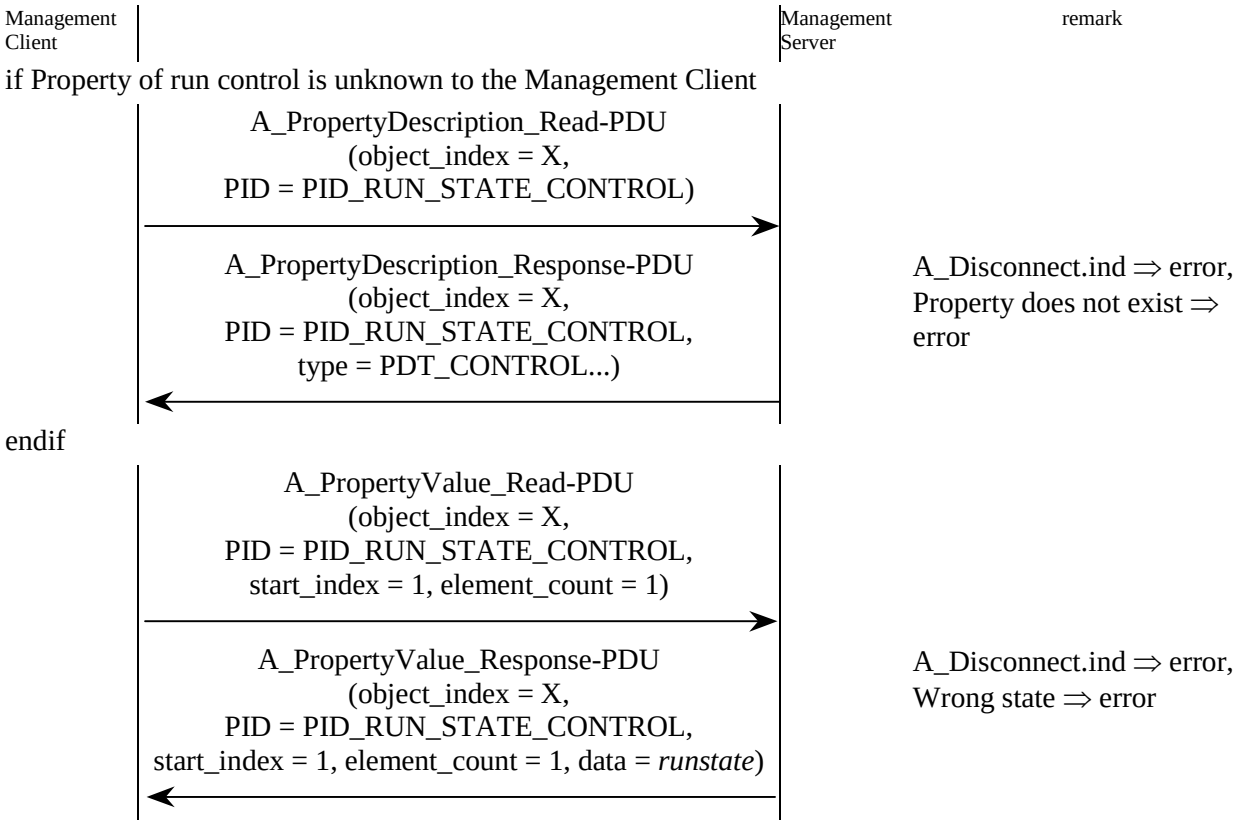
The control and state of the Run State Machine shall be located in Interface Objects of the Management Server and shall be accessible via Property services.

The Management Client shall search the according Interface Object in the Management Server.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

To be completed.

3.33 DM_RunStateMachineRead

3.33.1 Use

This device Management Procedure shall be used to read the state of a Run State Machine of a Management Server The state shall be located in the Management Procedure.

A DM_Connect shall be executed before executing this Management Procedure.

DM_RunStateMachineRead	(dataBlockStartAddress, flags, stateMachineType, stateMachineNr, state)
flags	bit 0: location of data 0: in data block 1: - All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are stored in the data block.
stateMachineType	type of the object that contains the state machine

state machine	type
application program	0003
PEI program	0004

stateMachineNr	index to the state machine. For this index only the state machines of this type are relevant. This index starts counting from 0.

3.33.2 Procedure: DMP_RunStateMachineRead_RCo_Mem

This Management Procedure shall use the connection oriented communication mode.
The control and state of the Run State Machine shall be located in the memory of the Management Server and shall be accessible as Memory Mapped Resource.

This Management Procedure shall support only one state machine of each type.

This Management Procedure shall only be used with device model for mask version 070nh (BIM M112).
The address (*AAAA*) of the run state depends on the Run State Machine.

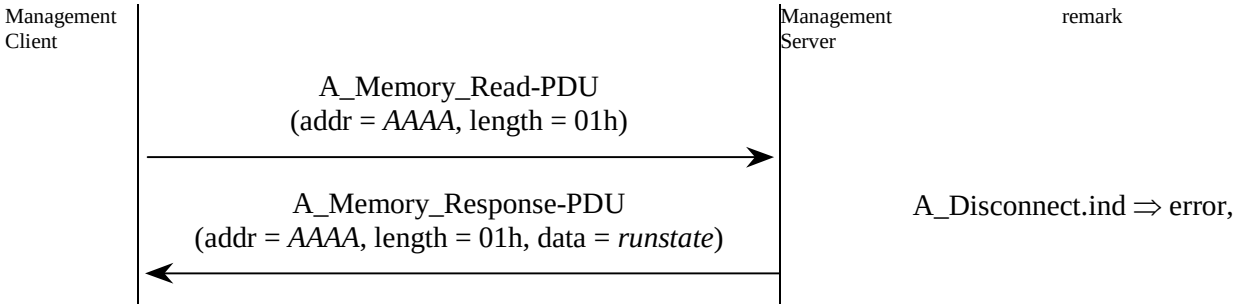
state machine	address of run state
application program	0101h
PEI program	0102h

This Management Procedure shall not be used for further developments of Management Servers.

Used Application Layer Services for Management

- A_Memory_Read

Sequence



Exception handling

To be completed.

3.33.3 Procedure: DMP_RunStateMachineRead_R_IO

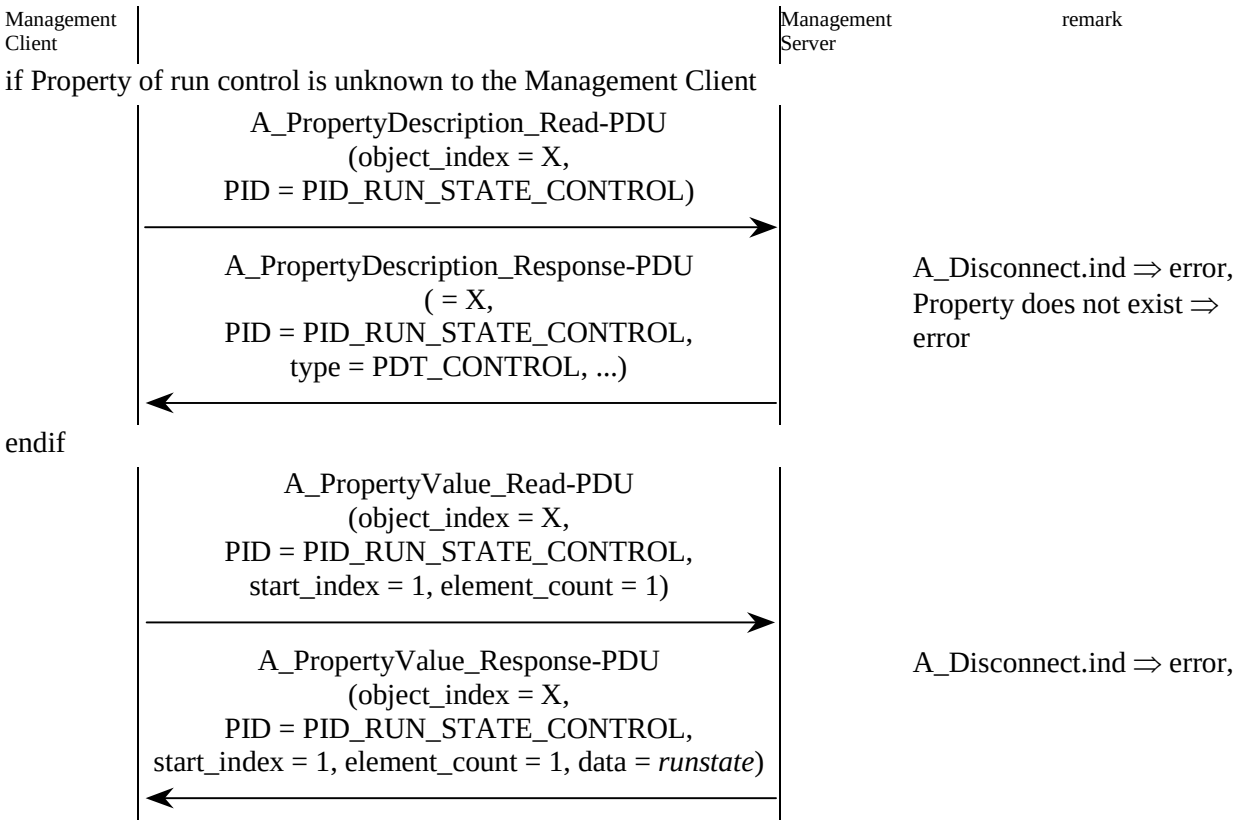
This method shall use either the connection oriented or the connectionless communication mode.
The control and state of the Run State Machine shall be located in Interface Objects of the Management Server and shall be accessible via Property services.

The Management Client shall search the according Interface Object in the Management Server.

Used Application Layer Services for Management

- A_PropertyDescription_Read
- A_PropertyValue_Read

Sequence



Exception handling

The general exception handling shall apply.

3.34 Procedures with Link Services

3.34.1 Basic requirements

To easily and quickly update a link between a Group Object in a device and a Group Address, independently of the target microcontroller, the A_Link_Read and A_Link_Write-services are provided.

This link establishment consists of a single link connecting a Group Object to a Group Address (this is, to a shared variable).

The Management Client assigning the new Group Addresses shall

- previously have checked that this Group Address is free, and
- know the Individual Address of the target device.

Requirements and error handling

Minimal Management Server side implementations may only support start_index 1. They shall react with a negative response (start_index = 0) when receiving an A_Link_Read-PDU or an A_Link_Write-PDU with any other start_index.

Any failure to execute a request at Network Management level (table full, non-existing group_object_number, invalid reserved bits, unknown start_index …) shall be answered by the negative response.

3.34.2 DM_GroupObjectLink_Read_RCl

Used Application Layer Services for Management

- A_Link_Read

Parameters of the Management Procedure

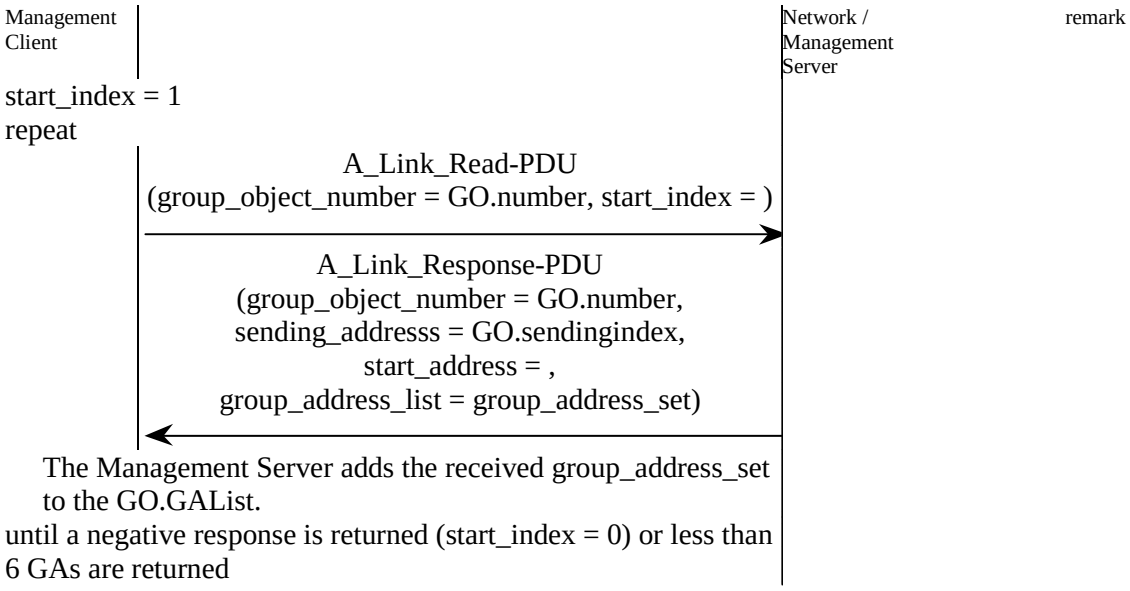
DM_GroupObjectLink_Read_RCl(/* [in] */ GO.number, /* [out] */ GO.sendingindex,
/* [out] */ GO.GAList)

GO.number:	Number of the Group Object from which the associated Group Addresses shall be read.
GO.sendingindex:	The index of the sending Group Address for this Group Object as returned by the Management Server (device).
GO.GAList	The list of all Group Addresses associated to the Group Object as returned by the Management Server (device)

Variables

start_index_loop:	The Management Client shall start reading the Group Addresses associated to the Group Object in the Management Server starting from the 1st one in the list, with start_index =start_index°loop 1. If an response is received from the Management Server with 6 Group Addresses, then the request shall be repeated: start_index_loop shall be incremented by 6 and used as start_index for the next call of the A_Link_Read-service by the Management Client
group_address_set:	A set of 6 or less GAs returned by the Management Server (device) in one A_Link_Response-PDU. The Management Client collects all these responses of one or more iterations in GO.GAList

Sequence



3.34.3 DM_GroupObjectLink_Write_RCl

Used Application Layer Services for Management

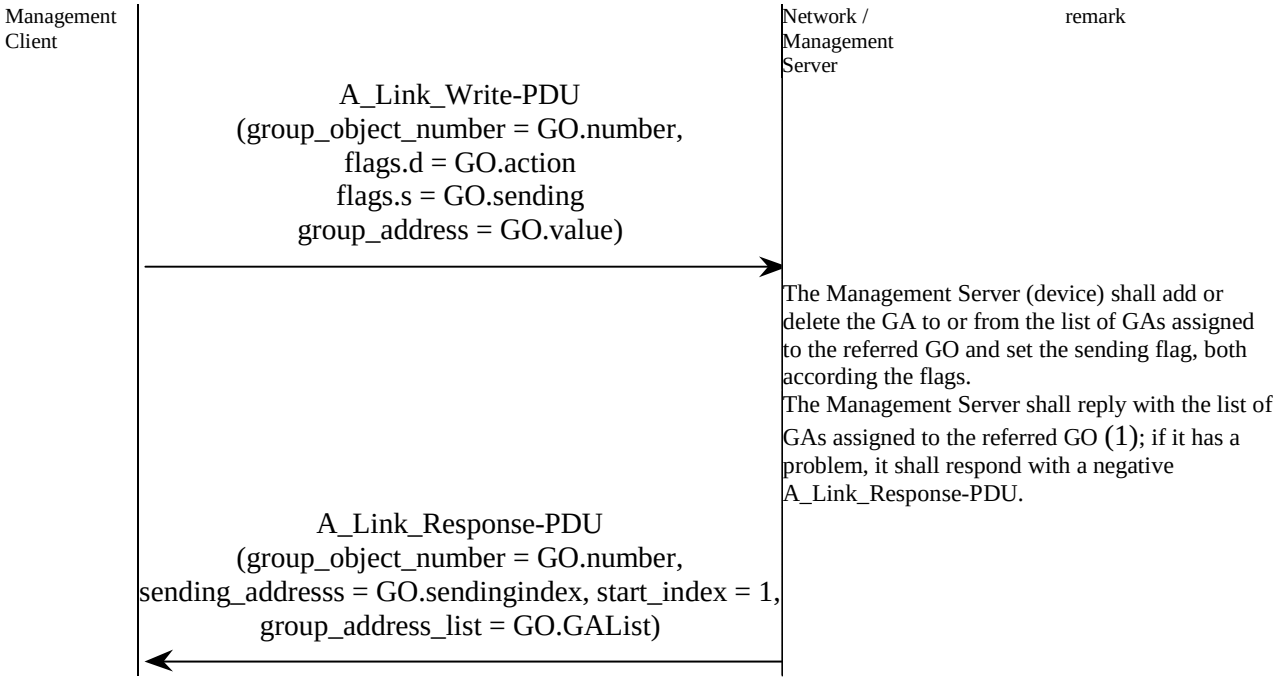
- A_Link_Write

Parameters of the Management Procedure

DM_GroupObjectLink_Write_RCl(/* [in] */ GO.number, /* [in] */ GO.action, /* [in] */ GO.sending, /* [in] */ GO.value, /* [out] */ GO.sendingindex, /* [out] */ GO.GAList)

GO.number:	Number of the GO to which the GA shall be added or from which the GA shall be removed.
GO.action	This flag shall indicate whether the contained GA shall be added to the GO or be removed from the GO. 0: add: The contained GA shall be added to the list of GA assigned to the referred GO. 1: delete: The contained GA shall be removed from the list of GAs assigned to the referred GO.
GO.sending:	Indication whether the added GA shall be the sending GA or not. This flag shall not be interpreted if the GA is removed.
GO.value:	The value of the GA hat shall be linked to or unlinked from the GO.
GO.sendingindex:	The index of the sending Group Address for this Group Object as returned by the Management Server (device).
GO.GAList	The list of all Group Addresses associated to the Group Object as returned by the Management Server (device).

Sequence



Error and exception handling

(1) The Management Server shall reply with the list of GAs assigned to the GO starting with start_index 1, thus with the first GA assigned to the GO. The number of GAs in the A_Link_Response-PDU shall be in-between 0 and 6, in function of the number of GAs assigned to the GO. If more than 6 GA are assigned, the Management Server shall only respond with the 6 first ones. The Management Server shall only send one single A_Link_Response-PDU. This means that the Management Client may take into account that the received A_Link_Response-PDY does not contain the GA that has just been added.

3.35 DM_LCSlaveMemWrite

3.35.1 Use

This device Management Procedure shall write a contiguous block of data to the specified memory addresses of the slave memory in the Management Server (Coupler). The data shall be located either in the management control or in the data block. Only the data that are specified in the data block are written. Depending on the flag the data shall be verified immediately. If the deviceStartAddress if higher than the deviceEndAddress, this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure is specific for the first generation Coupler model and shall not be used for further developments of Management Servers.

DM_LCSlaveMemWrite	(flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)
flags	bit 0: location of data 0: in data block 1: in Management Procedure bit 1: verify enabled / disabled 0: disabled 1: enabled All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress	address of first memory octet that is written by this Management Procedure
deviceEndAddress	address of the last memory octet that is written by this Management Procedure
data	the data that are transferred by this Management Procedure. The data can be located in the data block or in the Management Procedure.

3.35.2 Procedure: DMP_LCSlaveMemWrite_RCo

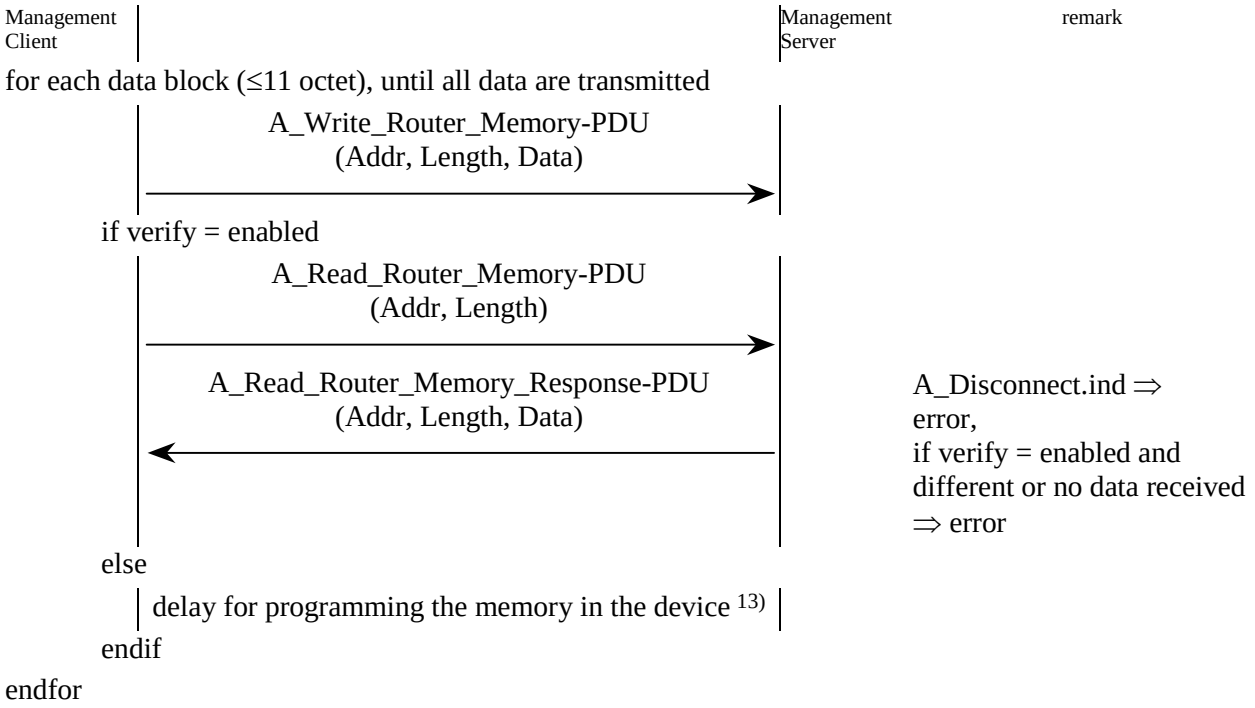
This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall not be used.

Used Application Layer Services for Management

- A_Write_Router_Memory
- A_Read_Router_Memory

Sequence



Exception handling

The general exception handling shall apply.

3.36 DM_LCSlaveMemVerify

3.36.1 Use

This device Management Procedure shall read a contiguous block of slave memory in the Management Server (Coupler) and compare it with the specified data. The data shall be located either in the management control or in the data block. Only the data that are specified in the data block shall be compared. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

13) The delay time depends on the Management Server and on the amount of written octets (see [08]).

DM_LCSlaveMemVerify	(flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)
flags	bit 0: location of data 0: in data block 1: in management control All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress	address of first memory octet that is compared by this Management Procedure
deviceEndAddress	address of the last octet that is compared by this Management Procedure
data	the data that are compared by this Management Procedure. The data can be located in the data block or in the Management Procedure.

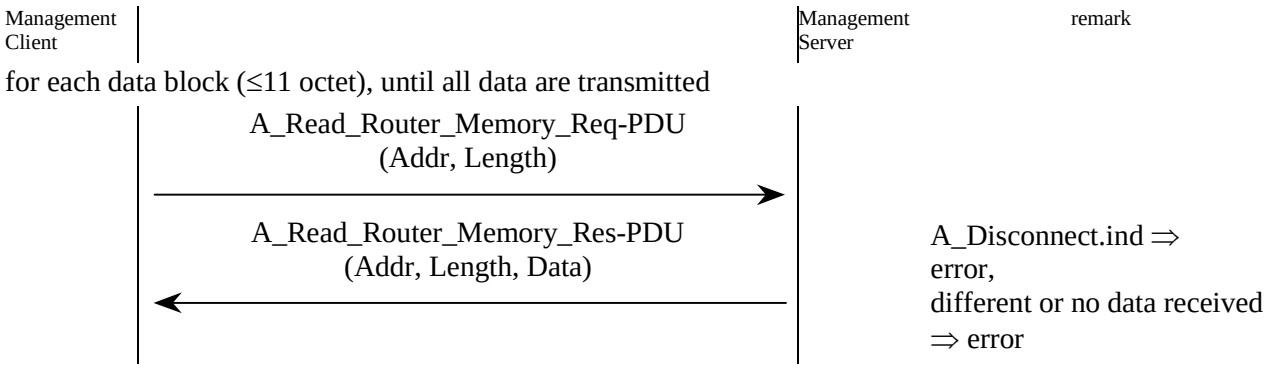
3.36.2 Procedure: DMP_LCSlaveMemVerify_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Read_Router_Memory

Sequence



Exception handling

The general exception handling shall apply.

3.37 DM_LCSlaveMemRead

3.37.1 Use

This device Management Procedure shall read a contiguous block of slave memory in the Management Server (Coupler) and store it in the data block. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

DM_LCSlaveMemRead (flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)

flags	bit 0	location of data 0: in data block 1: -
	All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.	
dataBlockStartAddress	specifies the address where the data are located in the data block.	
deviceStartAddress	address of first memory octet that is read by this Management Procedure	
deviceEndAddress	address of the last memory octet that is read by this Management Procedure	
data	the data that are read by this Management Procedure. The data are stored in the data block.	

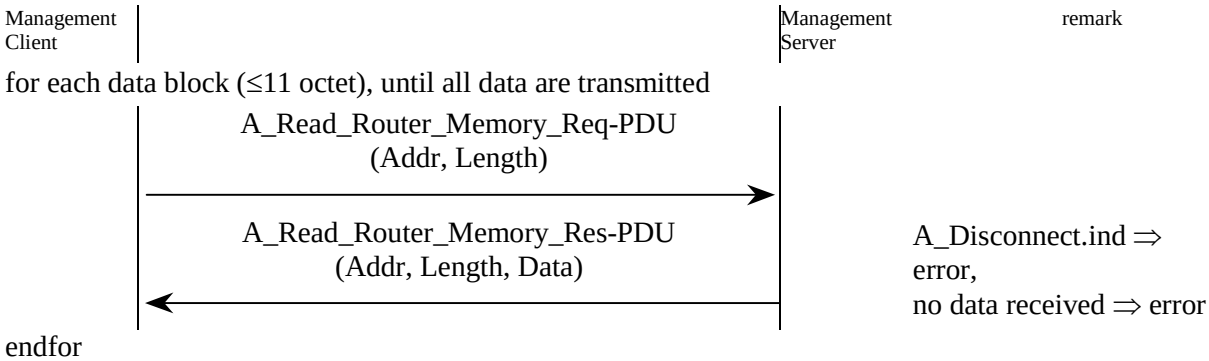
3.37.2 Procedure: DMP_LCSlaveMemRead_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Read_Router_Memory

Sequence



Exception handling

The general exception handling shall apply.

3.38 DM_LCExtMemWrite

3.38.1 Use

This device Management Procedure shall write a contiguous block of data to the specified memory addresses of the external memory in the Management Server (Coupler). The data shall be located either in the management control or in the data block. Only the data that are specified in the data block shall be written. Depending on the flag the data shall be verified immediately. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

DM_LCExtMemWrite		(flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)
flags	bit 0	location of data 0: in data block 1: in Management Procedure
	bit 1	verify enabled / disabled 0: disabled 1: enabled
	All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.	
dataBlockStartAddress	specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.	
deviceStartAddress	address of first memory octet that is written by this Management Procedure	
deviceEndAddress	address of the last memory octet that is written by this Management Procedure	
data	the data that are transferred by this Management Procedure. The data can be located in the data block or in the Management Procedure.	

3.38.2 Procedure: DMP_LCExtMemWrite_RCo

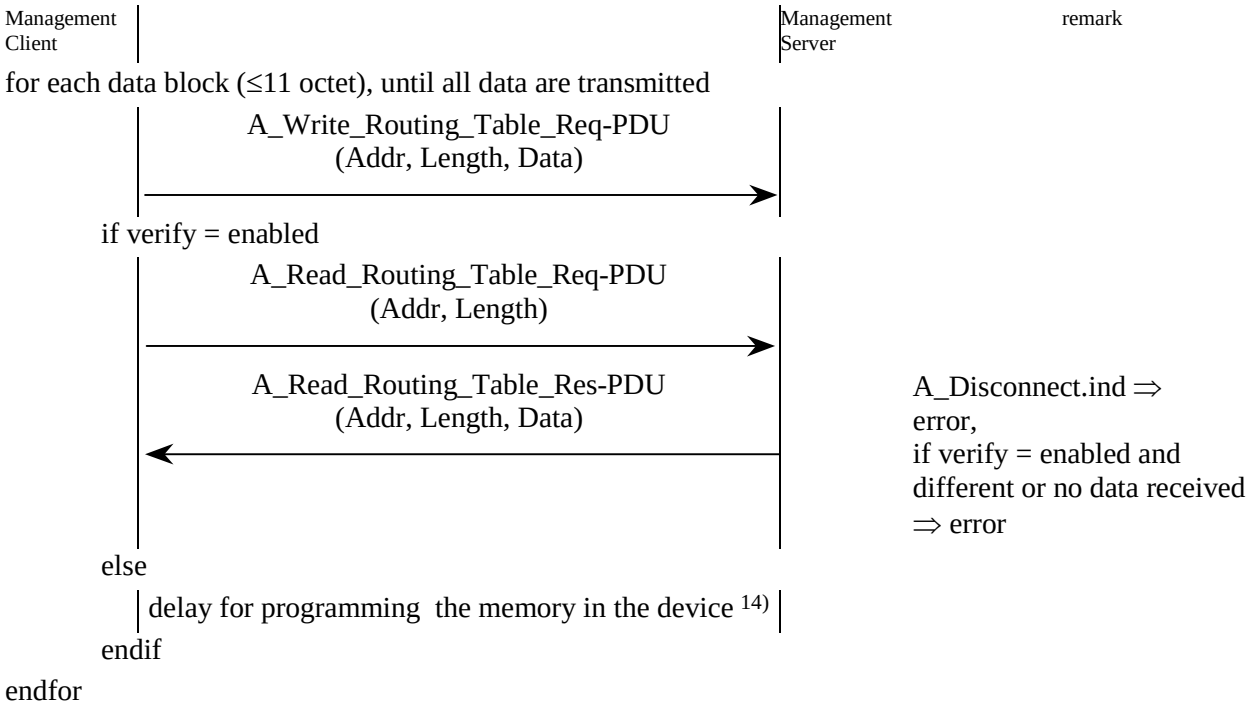
This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall not be used.

Used Application Layer Services for Management

- A_Write_Routing_Table
- A_Read_Routing_Table

Sequence



Exception handling

The general exception handling shall apply.

3.39 DM_LCExtMemVerify

3.39.1 Use

This device Management Procedure shall read a contiguous block of external memory from the Management Server (Coupler) and compare it with the specified data. The data shall be located either in the management control or in the data block. Only the data that are specified in the data block shall be compared. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

14) The delay time depends on the Management Server and on the amount of written octets (see [08]).

DM_LCExtMemVerify		(flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)
flags	bit 0	location of data 0: in data block 1: in management control
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress		specifies the address where the data are located in the data block. If the data are located in the Management Procedure, this field is set to 0.
deviceStartAddress		address of first memory octet that is compared by this Management Procedure
deviceEndAddress		address of the last memory octet that is compared by this Management Procedure
data		the data that are compared by this Management Procedure. The data can be located in the data block or in the Management Procedure.

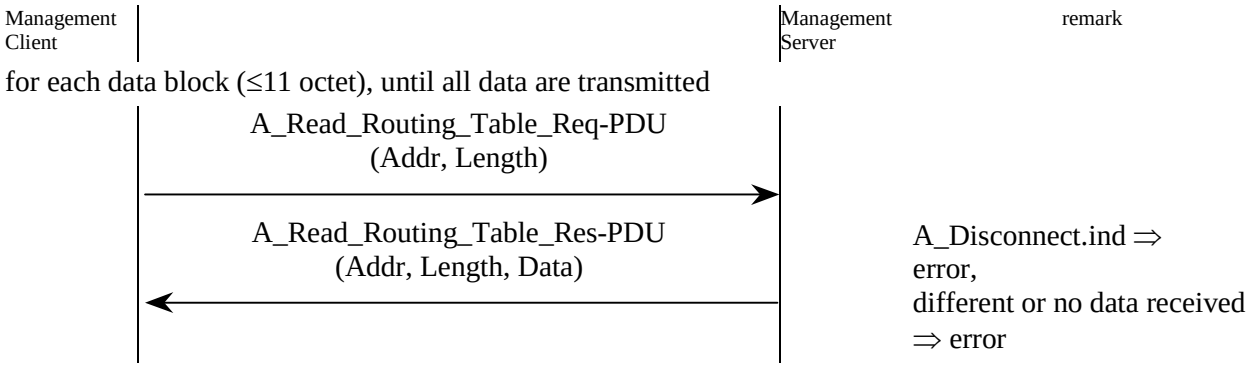
3.39.2 Procedure: DMP_LCExtMemVerify_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Read_Routing_Table

Sequence



Exception handling

The general exception handling shall apply.

3.40 DM_LCExtMemRead

3.40.1 Use

This device Management Procedure shall read a contiguous block of external memory from the Management Server (Coupler) and store it in the data block. If the deviceStartAddress if higher than the deviceEndAddress this Management Procedure shall be skipped.

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

DM_LCExtMemRead (flags, dataBlockStartAddress, deviceStartAddress, deviceEndAddress, data)

flags	bit 0	location of data 0: in data block 1: -
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
dataBlockStartAddress		specifies the address where the data are located in the data block.
deviceStartAddress		address of first memory octet that is read by this Management Procedure
deviceEndAddress		address of the last memory octet that is read by this Management Procedure
data		the data that are read by this Management Procedure. The data are stored in the data block.

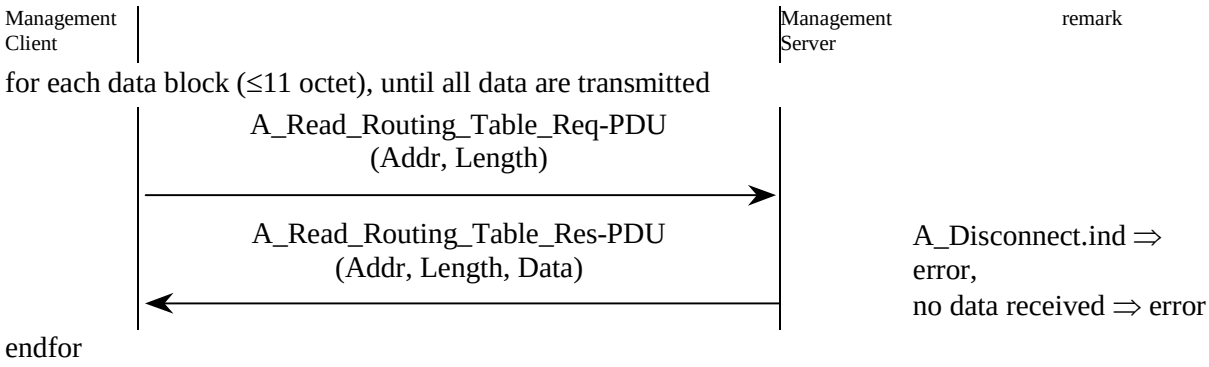
3.40.2 Procedure: DMP_LCExtMemRead_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Read_Routing_Table

Sequence



Exception handling

The general exception handling shall apply.

3.41 DM_LCExtMemOpen

3.41.1 Use

This device Management Procedure shall be used to enable writing in the external memory of the Management Server (Coupler).

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

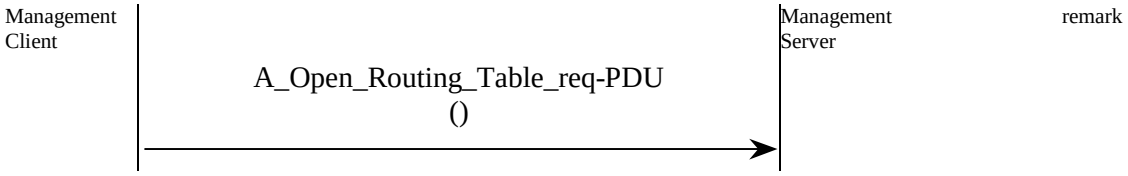
3.41.2 Procedure: DMP_LCExtMemOpen_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Open_Routing_Table

Sequence



Exception handling

The general exception handling shall apply.

3.42 DM_LCRouteTableStateWrite

3.42.1 Use

This device Management Procedure shall be used to write the state of the routing table to the Management Server (line coupler).

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

DM_LCRouteTableStateWrite (flags, routeTableState)

flags	bit 0	location of data (routeTableState) 0: in data block 1: in management control
	bit 1	verify enabled / disabled 0: disabled 1: enabled
	All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.	
routeTableState	state of the routing table (see description of APDU)	

3.42.2 Procedure: DMP_LCRouteTableStateWrite_RCo

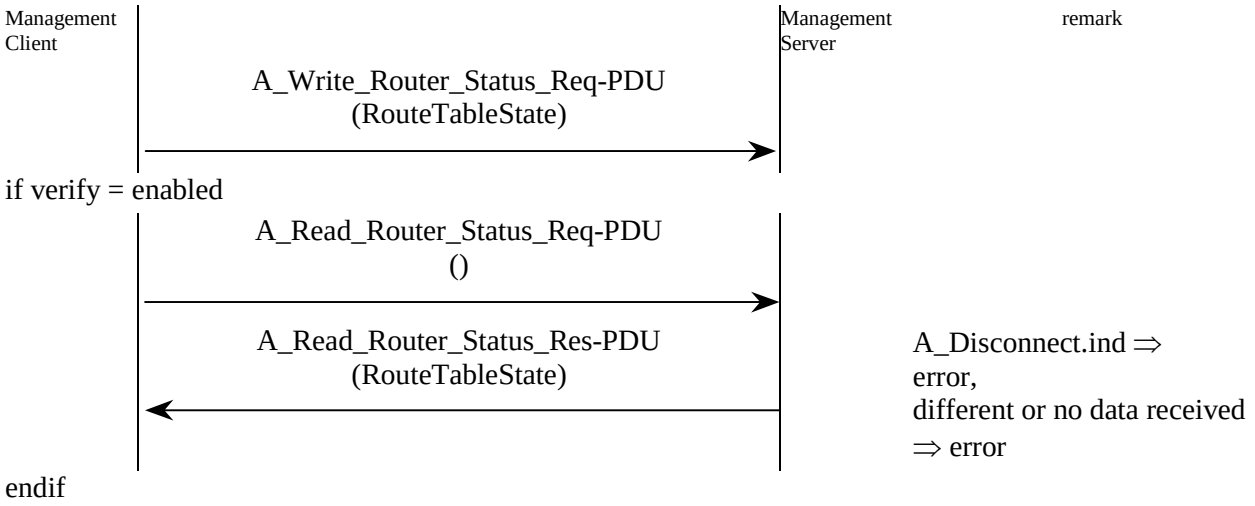
This Management Procedure shall use the connection oriented communication mode.

The Verify Mode of the Management Server shall not be used.

Used Application Layer Services for Management

- A_Write_Router_Status
- A_Read_Router_Status

Sequence



Exception handling

The general exception handling shall apply.

3.43 DM_LCRouteTableStateVerify

3.43.1 Use

This device Management Procedure shall be used to read and verify the state of the routing table to the Management Server (Coupler).

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

DM_LCRouteTableStateVerify (flags, routeTableState)

flags	bit 0	location of data (routeTableState) 0: in data block 1: in management control
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
routeTableState		state of the routing table (see description of APDU)

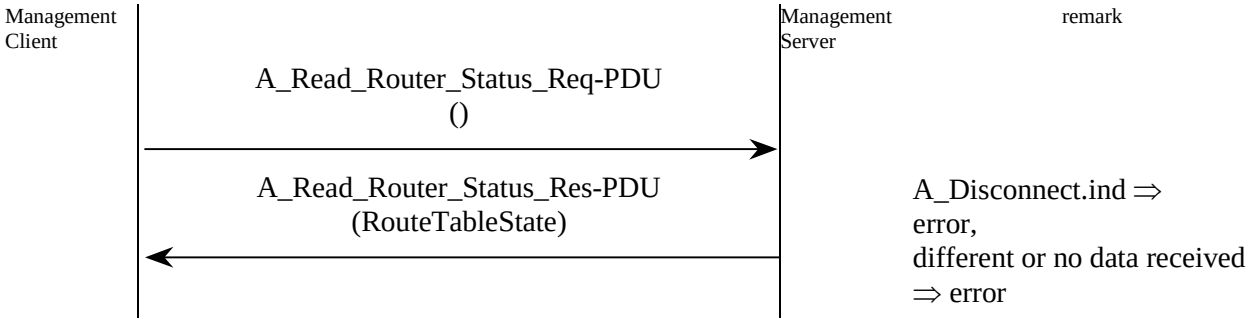
3.43.2 Procedure: DMP_LCRouteTableStateVerify_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Read_Router_Status_Req

Sequence



Exception handling

The general exception handling shall apply.

3.44 DM_LCRouteTableStateRead

3.44.1 Use

This device Management Procedure shall be used to read the state of the Routing Table to the Management Server (Coupler).

A DM_Connect shall be executed before executing this Management Procedure.

This device Management Procedure shall not be used for further developments of Management Servers.

DM_LCRouteTableStateRead (flags, routeTableState)

flags	bit 0	location of data (routeTableState) 0: in data block 1: -
		All other bits are reserved. These shall be set to 0. This shall be tested by the Management Client.
routeTableState	state of the routing table (see description of APDU)	

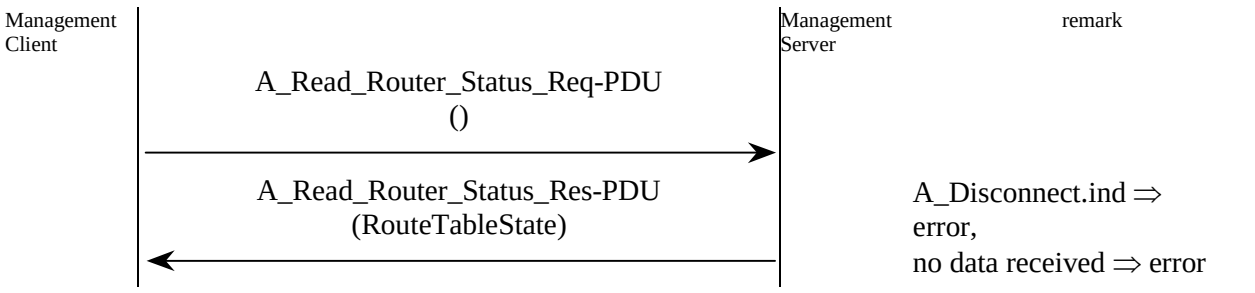
3.44.2 Procedure: DMP_LCRouteTableStateRead_RCo

This Management Procedure shall use the connection oriented communication mode.

Used Application Layer Services for Management

- A_Read_Router_Status

Sequence



Exception handling

The general exception handling shall apply.

4 KNXnet/IP Management Procedures

4.1 DMP_KNXnet/IP_Connect

This procedure shall be used to connect to a KNXnet/IP server using a specific connection type.

DMP_KNXnet/IP_Connect(/* [in] */ dmp_HPAIControlEndpoint, /* [in] */ dmp_HPAIDataEndpoint,
/* [in] */ dmp_CRI, /* [out] */ dmp_CommChannelID,
/* [out] */ dmp_Status, / *[out] */ dmp_HPAIServerDataEndpoint,
/ *[out] */ dmp_CRD)

Parameters of the Management Procedure

dmp_HPAIClientControlEndpoint:	This Management Procedure Parameter shall contain the HPAI of the Control Endpoint of the Management Client.
dmp_HPAIClientDataEndpoint:	This Management Procedure Parameter shall contain the HPAI of the Data Endpoint of the Management Client.
dmp_CRI:	This Management Procedure Parameter shall contain the Connection Request Information from the Management Client.
dmp_CommChannelID:	This Management Procedure Parameter shall return the communication channel ID that is chosen by the Management Server.
dmp_Status:	This Management Procedure Parameter shall return the status of the request.
dmp_HPAIServerDataEndpoint:	This Management Procedure Parameter shall contain the HPAI of the Data Endpoint of the Management Server.
dmp_CRD:	This Management Procedure Parameter shall contain the Connection Response Data Block from the Management Server.

Variables

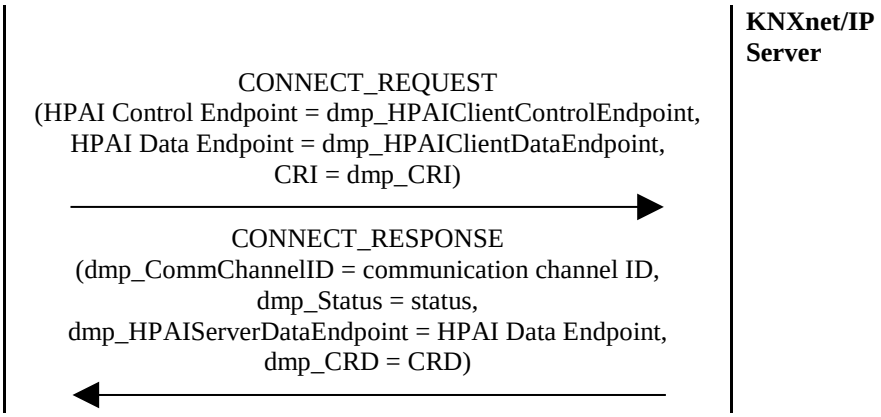
None.

Precondition

Through KNXnet/IP Discovery, it is confirmed that the KNXnet/IP device supports the requested connection type.

NOTE 8 To this, the KNXnet/IP Client sends a DESCRIPTION_REQUEST to the control endpoint of the device to which it wants to establish a KNXnet/IP Tunnelling Connection. The device shall respond with a DESCRIPTION_RESPONSE frame holding the “DIB supported service families”. If the device supports KNXnet/IP Tunnelling, one of the DIBs will report a Supported Service family 04h, this is, KNXnet/IP Tunnelling.

Sequence



4.2 DMP_InterfaceObjectWrite_IP

Used KNXnet/IP Services

- DEVICE_CONFIGURATION_REQUEST
  with cEMI-services
  - M_PropWrite.req
  - M_PropWrite.con
- DEVICE_CONFIGURATION_ACK

Preconditions

It is assumed that any type of KNXnet/IP connection exists to the Management Server with Communication Channel ID dmp_CommChannelID.

Parameters of the Management Procedure

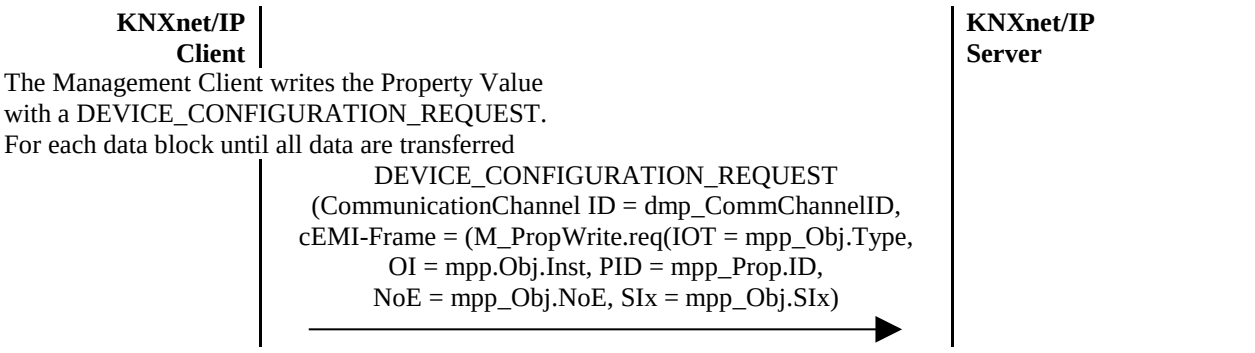
DMP_InterfaceObjectWrite_IP(/* [in] */ mpp_Obj.Type, /* [in] */ mpp_Obj.Inst, /* [in] */ mpp_Prop.ID, /* [in] */ mpp_Prop.Value, /* [in] */ mpp_Prop.StartIndex, /* [in] */ = mpp_Prop.NrOfElem, /* [out] */ = mpp_ErrorCode)

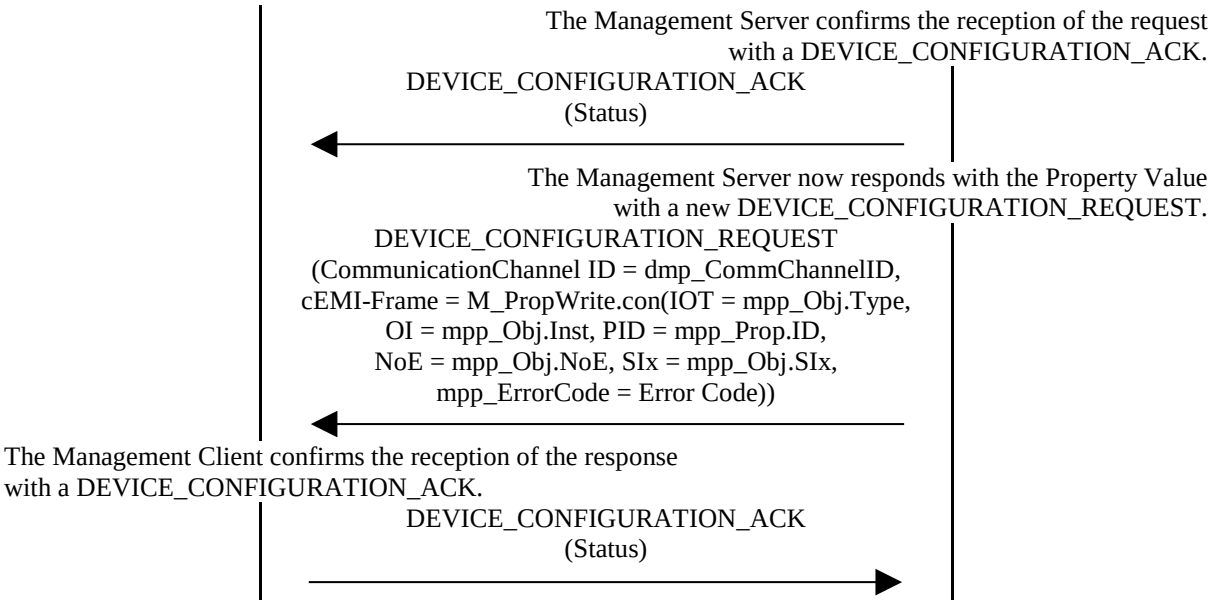
mpp_Obj.Type:	This Management Procedure Parameter shall hold the Object Type of the Interface Object in which the Property shall be written.
mpp_Obj.Inst:	This Management Procedure Parameter shall hold the Instance of the Interface Object in which the Property shall be written.
mpp_Prop.ID:	This Management Procedure Parameter shall hold the Property Identifier of the Property that shall be written.
mpp_Prop.Value:	This Management Procedure Parameter shall hold the value of the Property that is shall be written.
mpp_Prop.StartIndex:	This Management Procedure Parameter shall hold the start index within the Property value from which index onwards the Property value shall be written.
mpp_Prop.NrOfElem:	This Management Procedure Parameter shall hold the number of Property array elements that shall be written.
mpp_ErrorCode:	This Management Procedure Parameter shall return the Error Code that may be returned by the Management Server.

Variables

mpp_Obj.NoE	This shall hold the number of elements of the Property Value that is written in one call of the cEMI service M_PropWrite.req.
mpp_Obj.SIx:	This shall hold the start index within the Property Value from which the Property elements are written.

Sequence





4.3 DMP_InterfaceObjectRead_IP

Used KNXnet/IP Services

- DEVICE_CONFIGURATION_REQUEST
  with cEMI-services
  - M_PropRead.req
  - M_PropRead.con
- DEVICE_CONFIGURATION_ACK

Preconditions

It is assumed that any type of KNXnet/IP connection exists to the Management Server with Communication Channel ID dmp_CommChannelID.

Parameters of the Management Procedure

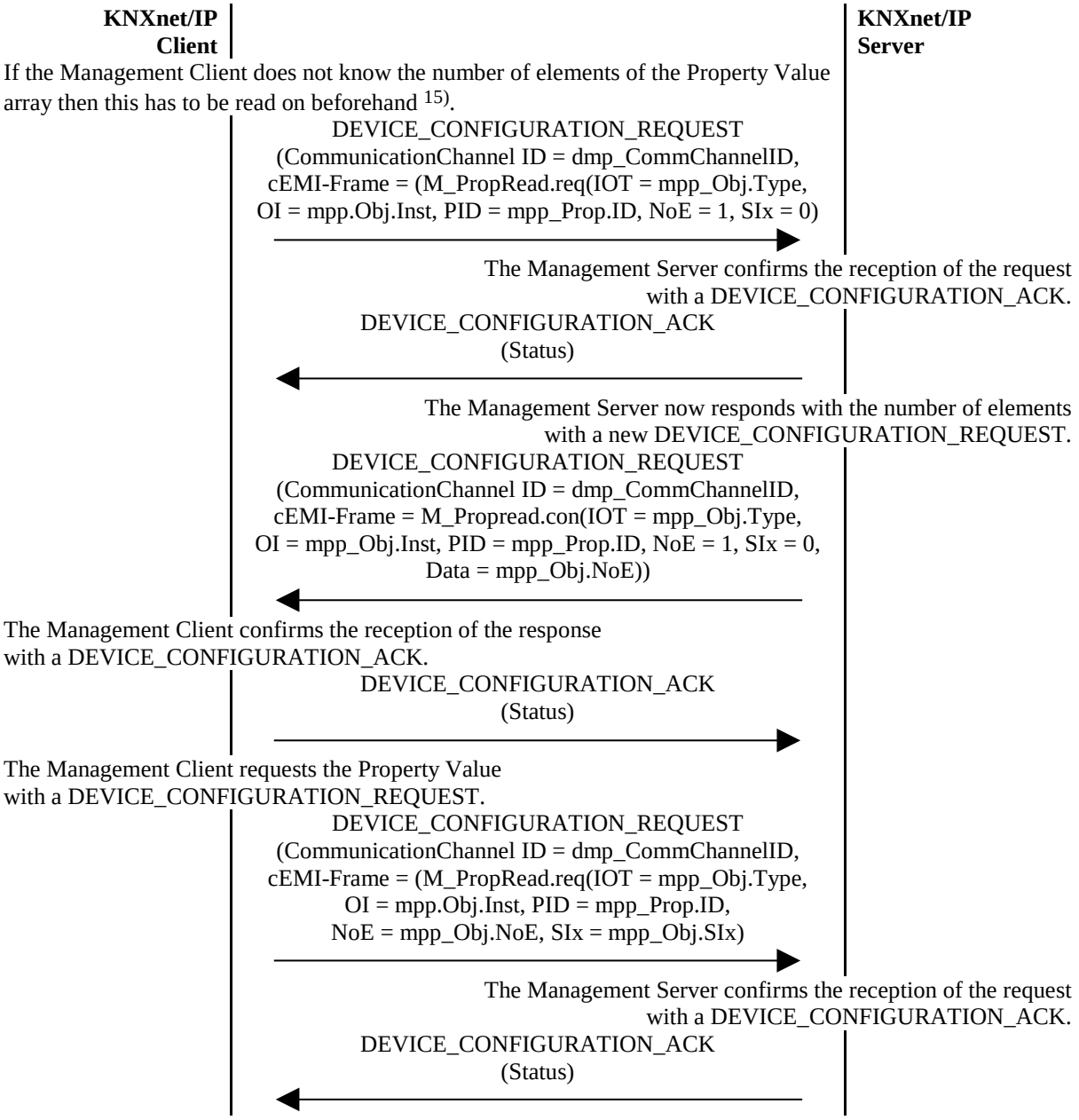
DMP_InterfaceObjectRead_IP(/* [in] */ mpp_Obj.Type, /* [in] */ mpp_Obj.Inst, /* [in] */ mpp_Prop.ID, /* [out] */ mpp_Prop.Value, /* [out] */ mpp_Prop.CurrentNr)

mpp_Obj.Type:	This Management Procedure Parameter shall hold the Object Type of the Interface Object in which the Property shall be read.
mpp_Obj.Inst:	This Management Procedure Parameter shall hold the Instance of the Interface Object in which the Property shall be read.
mpp_Prop.ID:	This Management Procedure Parameter shall hold the Property Identifier of the Property that shall be read.
mpp_Prop.Value:	This Management Procedure Parameter shall return the value of the Property that is read.
mpp_Prop.CurrentNr:	This Management Procedure Parameter shall return the current number of elements of the Property that is read.

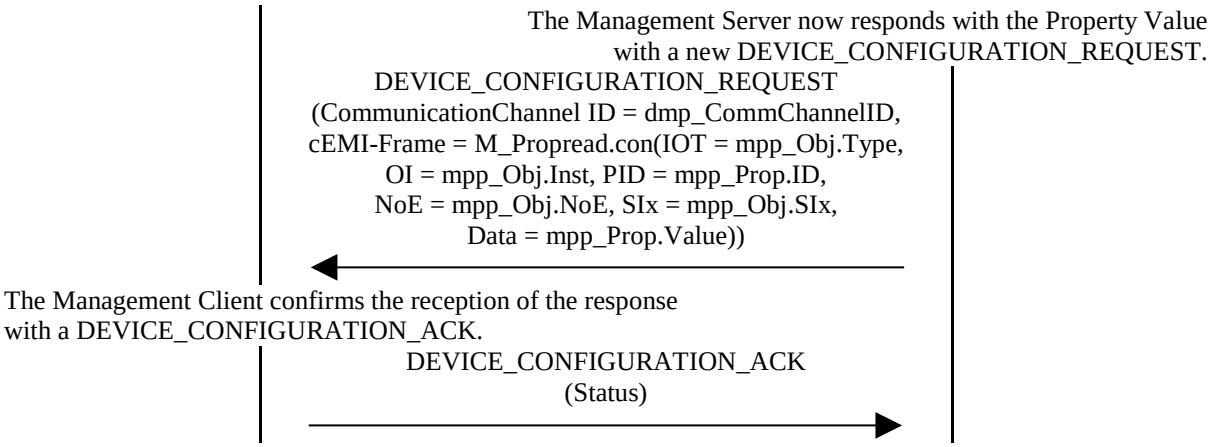
Variables

mpp_Obj.NoE	This shall hold the number of elements of the Property Value that is read in one call of the cEMI service M_PropRead.req.
mpp_Obj.SIx:	This shall hold the start index within the Property Value from which the Property elements are read.

Sequence



15) This first reading is thus optional. It should only occur if the Management Client does not know the number of elements of the Property Value. Specifically, this should be done when the Property Value may be an array of which the current number of elements is unknown. By this value and by the knowledge of the maximal Frame length in-between the Management Client and the Management Server (see “Discovery of maximal Frame length” in [4]), the Management Client may know how many read operations may be required and how much data can fit in one single response.


The Management Server now responds with the Property Value
with a new DEVICE_CONFIGURATION_REQUEST.
DEVICE_CONFIGURATION_REQUEST
(CommunicationChannel ID = dmp_CommChannelID,
cEMI-Frame = M_Propread.con(IOT = mpp_Obj.Type,
OI = mpp_Obj.Inst, PID = mpp_Prop.ID,
NoE = mpp_Obj.NoE, SIx = mpp_Obj.SIx,
Data = mpp_Prop.Value))
The Management Client confirms the reception of the response
with a DEVICE_CONFIGURATION_ACK.
DEVICE_CONFIGURATION_ACK
(Status)

5 File Transfer Procedures

5.1 Preconditions and error handling

5.1.1 Preconditions

he Management Client shall act as a File Client. If the File Server Object is unknown to the Management Client, then before accessing the File Server Object the Management Client shall use DM_InterfaceObjectScan to find the Object Index of the File Server Object(s).

5.1.2 Common error and exception handling

For all procedures that transfer a file or a directory listing, the corresponding receiver shall stop waiting for data after a certain time-out (if no data arrive). If a time-out occurs, an error message shall be created.

Procedures that convey an invalid command shall be processed normally. The command shall simply be ignored and the appropriate error code shall be returned. This error code has influence to the next steps of the procedure.

5.2 FTP_RetrieveFile

Use

This Management Procedure shall be used to read a file in FTP (raw) mode from a File Server.

Used Application Layer messages for management

- A_FunctionPropertyCommand-PDU(destination_address, object_index, Property_id, data)
- A_FunctionPropertyState_Response-PDU(destination_address, object_index, Property_id, return_code, data)
- A_PropertyValue_Write-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)
- A_PropertyValue_Response-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)
- A_FileStream_InfoReport-PDU(destination_address, file_handle, file_block_sequence_number, file_block)

Variables in FTP_RetrieveFile

Server IA	Individual Address of the FTP Server
File Server OI	Object Index of the File Server Object in the Management Server.
Client IA	Individual Address of the FTP Client.
File Path	The path to the file to be retrieved from the File Server.
File handle	File handle retrieved from the FTP server

Sequence

Management Client		Management Server	remark
repeat			
get file handle from File Server Object			
	A_FunctionPropertyCommand-PDU (Server IA, File Server OI, PID_FILE_COMMAND, data = Get File handle) →		The File Server shall execute the command “Get File handle” and return the File handle.
	← A_FunctionPropertyState_Response-PDU (Client IA, File Server OI, PID_FILE_COMMAND, Return Code, Return Value)		
If the Return Code is “Object busy”, take the Return Value as Object Index of the next free File Server Object. If this object index is > 0, set File Server OI to this value and repeat, else report “Server Busy”.			
until a valid file handle is received or no more File Server Objects are available.			
if the Return Code is “Command Successful”			
File handle = Return Value			
write file path into File Server Object Property PID_FILE_PATh (this needs not to be done if the file path is transferred within the File Command).			
	DMP_InterfaceObject_Write_R (Server IA, File Server OI, PID_FILE_PATh) →		
	A_FunctionPropertyCommand-PDU (Server IA, File Server OI, PID_FILE_COMMAND, data = Retrieve File) →		
	← A_FunctionPropertyState_Response-PDU (Client IA, File Server OI, PID_FILE_COMMAND, Return Code)		
if the Return Code is “Command Successful”, a file exists and will be transferred to the client			
			repeat
			The Management Server (File Server) shall transmit the file in consecutive A_FileStream_InfoReport-PDUs. Each APDU shall have an APDU-length of 14 or longer as given in PID_MAX_APDU_LENGTh. The last APDU may have a shorter length.
	← A_FileStream_InfoReport-PDU (Client IA, file handle, file_block_sequence_number, file_block)		
			until the whole file is transmitted mark end of file
			The field file_block is not present.
	← A_FileStream_InfoReport-PDU (Client IA, file handle, file_block_sequence_number)		
else if the Return Code is an Error Code, report this error.			
The File handle shall be released.			

5.3 FTP_StoreFile

Use

This Management Procedure shall be used to write a file in FTP (raw) mode to a File Server.

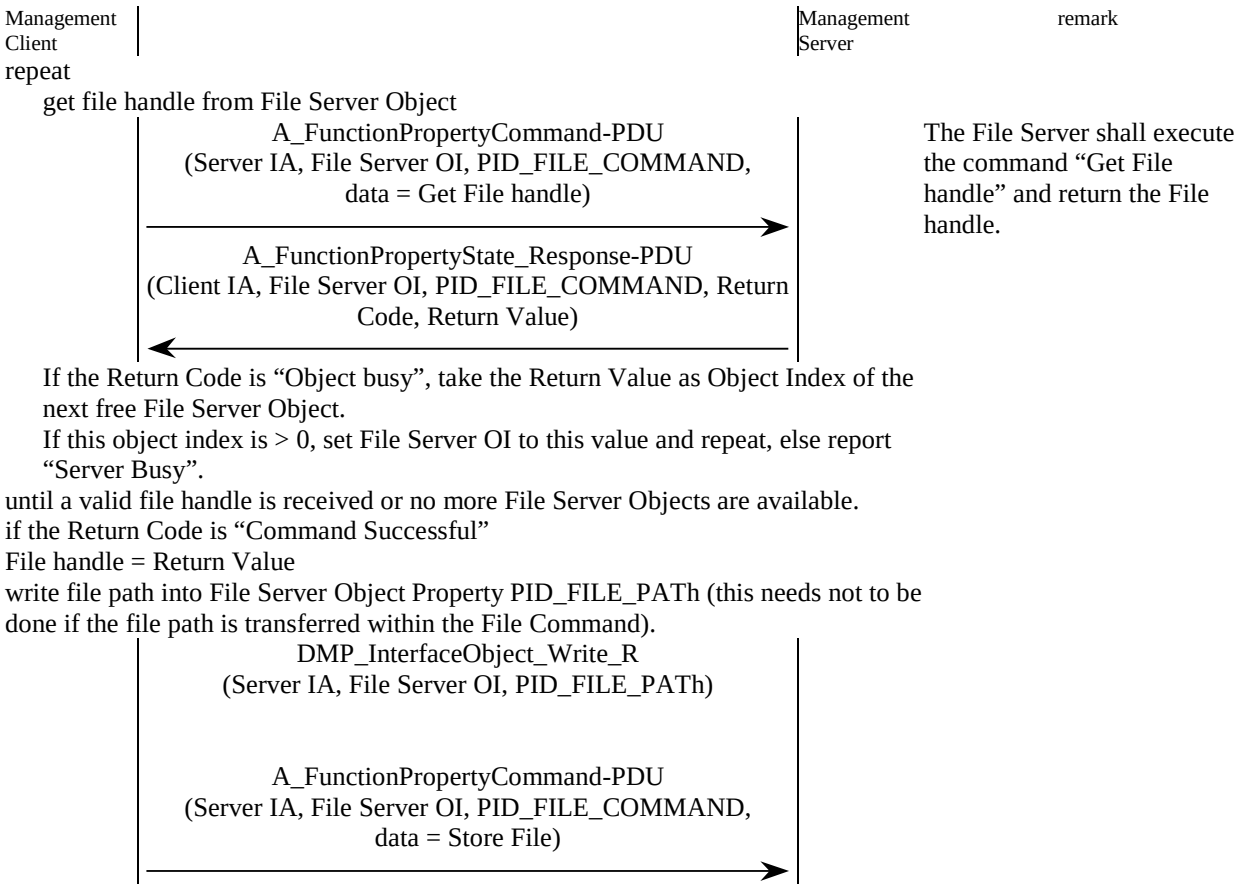
Used Application Layer messages for management

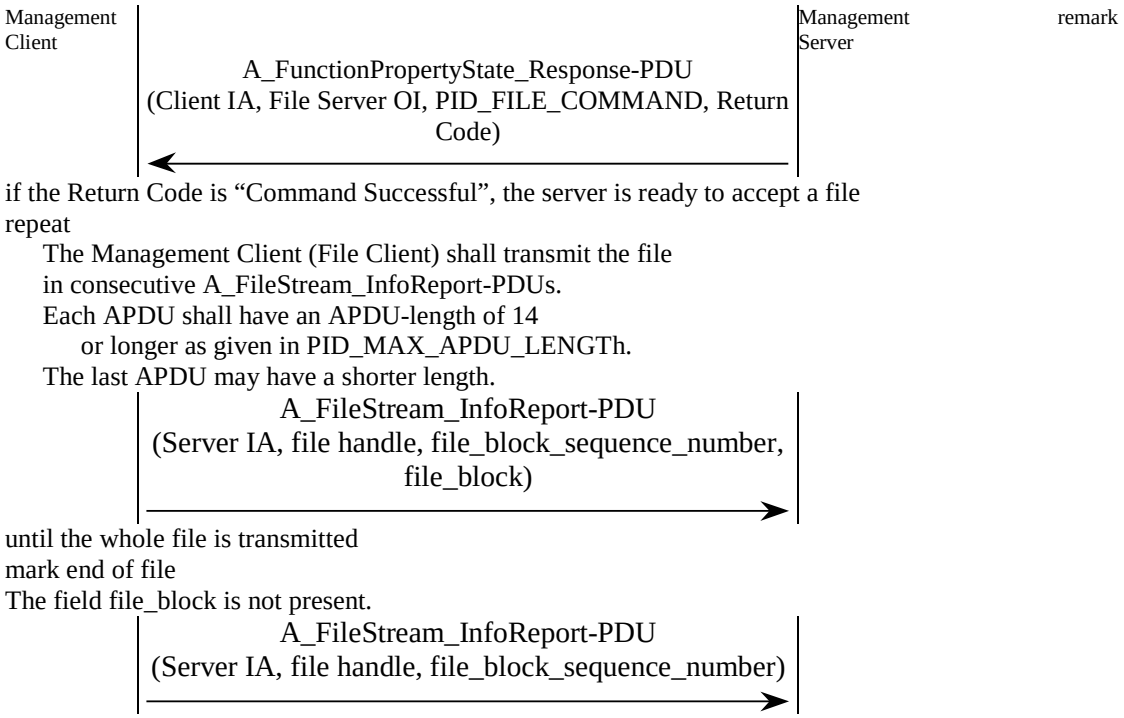
- A_FunctionPropertyCommand-PDU(destination_address, object_index, Property_id, data)
- A_FunctionPropertyState_Response-PDU(destination_address, object_index, Property_id, return_code, data)
- A_PropertyValue_Write-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)
- A_PropertyValue_Response-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)
- A_FileStream_InfoReport-PDU(destination_address, file_handle, file_block_sequence_number, file_block)

Variables in FTP_StoreFile

Server IA	Individual Address of the FTP Server
File Server OI	Object Index of the File Server Object in the Management Server.
Client IA	Individual Address of the FTP Client.
File Path	The path to the file to be stored in the File Server.
File handle	File handle retrieved from the FTP server

Sequence





5.4 FTP_ListDirectory

Use

This Management Procedure shall be used to read a directory listing from a File Server.

The procedure shall be identical to 5.2 “FTP_RetrieveFile”.

Every line of the directory listing shall start with a new A_FileStream_InfoReport-PDU. The number of octets transferred in a A_FileStream_InfoReport-PDU shall be either 14 or as specified in PID_MAX_APDU_LENGTh_OUT. It is although allowed that less octets than the maximum number are transferred. This makes it possible to start every directory line with a new A_FileStream_InfoReport-PDU.

5.5 FTP_Rename (consisting of Rename From and Rename To)

Use

This Management Procedure shall be used to rename a file on a File Server. The procedure consists of a sequence of the commands Rename From and Rename To. The File handle requested from the File Server for the Rename From command shall also be used for the Rename To command i.e. no extra File handle is requested for the Rename To command. The File handle shall therefore not be released after the execution of the Rename From command. It shall be released after the execution of the Rename To command (or after the time-out of 6 s, if one or both commands are missing).

This is to prevent the execution of a command other than Rename To after a Rename From coming from another File Client.

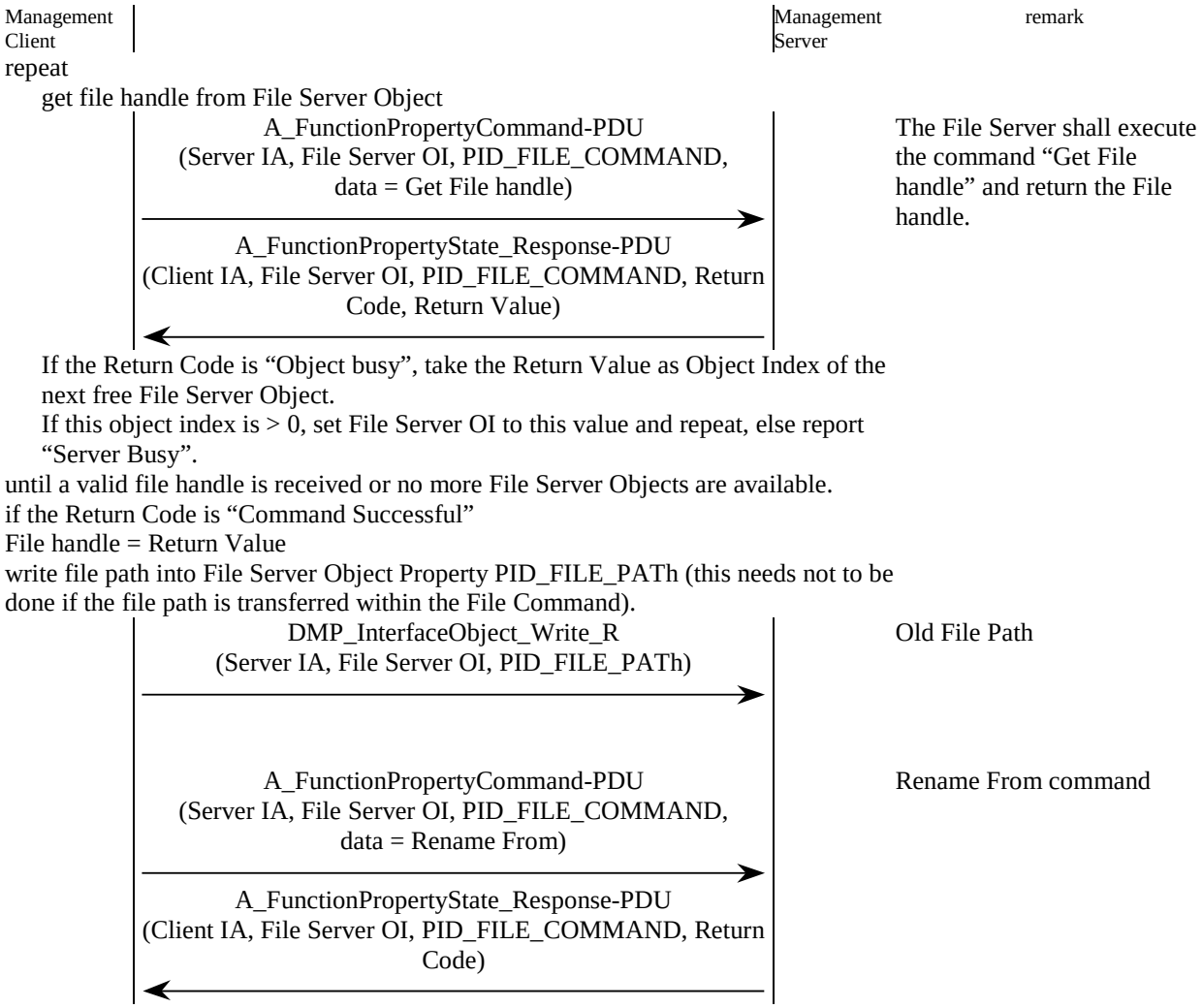
Used Application Layer messages for management

- A_FunctionPropertyCommand-PDU(destination_address, object_index, Property_id, data)
- A_FunctionPropertyState_Response-PDU(destination_address, object_index, Property_id, return_code, data)
- A_PropertyValue_Write-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)
- A_PropertyValue_Response-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)

Variables in FTP_Rename

Server IA	Individual Address of the FTP Server
File Server OI	Object Index of the File Server Object in the Management Server.
Client IA	Individual Address of the FTP Client.
File Path	The path to the file to be renamed from the File Server.
File handle	File handle retrieved from the FTP server

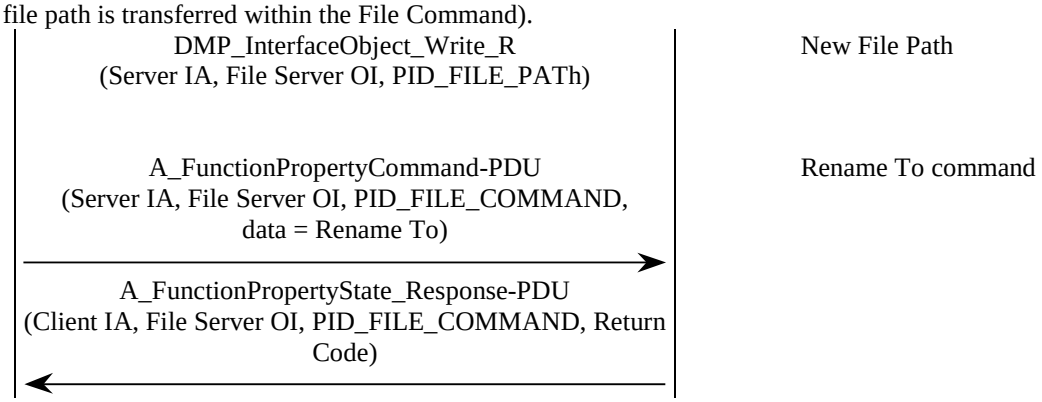
Sequence



Management Client		Management Server	remark

if the Return Code is “Command Successful”, the command Rename To shall be executed
else an error shall be reported.

write file path into File Server Object Property PID_FILE_PATh (this needs not to be done if the file path is transferred within the File Command).



if the Return Code is “Command Successful”, the command was executed correctly
else an error shall be reported.
The File handle shall be released.

5.6 FTP_Delete

Use

This Management Procedure shall be used to delete a file on a File Server.

Used Application Layer messages for management

- A_FunctionPropertyCommand-PDU(destination_address, object_index, Property_id, data)
- A_FunctionPropertyState_Response-PDU(destination_address, object_index, Property_id, return_code, data)
- A_PropertyValue_Write-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)
- A_PropertyValue_Response-PDU(destination_address, object_index, Property_id, nr_of_elem, start_index, data)

Variables in FTP_Delete

Server IA	Individual Address of the FTP Server
File Server OI	Object Index of the File Server Object in the Management Server.
Client IA	Individual Address of the FTP Client.
File Path	The path to the file to be deleted from the File Server.
File handle	File handle retrieved from the FTP server

Sequence

Management Client		Management Server	remark
repeat			
get file handle from File Server Object			
	A_FunctionPropertyCommand-PDU (Server IA, File Server OI, PID_FILE_COMMAND, data = Get File handle) →		The File Server shall execute the command “Get File handle” and return the File handle.
	← A_FunctionPropertyState_Response-PDU (Client IA, File Server OI, PID_FILE_COMMAND, Return Code, Return Value)		
If the Return Code is “Object busy”, take the Return Value as Object Index of the next free File Server Object. If this object index is > 0, set File Server OI to this value and repeat, else report “Server Busy”.			
until a valid file handle is received or no more File Server Objects are available.			
if the Return Code is “Command Successful”			
File handle = Return Value			
write file path into File Server Object Property PID_FILE_PATh (this needs not to be done if the file path is transferred within the File Command).			
	DMP_InterfaceObject_Write_R (Server IA, File Server OI, PID_FILE_PATh)		
	A_FunctionPropertyCommand-PDU (Server IA, File Server OI, PID_FILE_COMMAND, data = Delete) →		
	← A_FunctionPropertyState_Response-PDU (Client IA, File Server OI, PID_FILE_COMMAND, Return Code)		
if the Return Code is “Command Successful”, the command was executed correctly else an error shall be reported.			
The File handle shall be released.			

5.7 FTP_RemoveDirectory

Use

This Management Procedure shall be used to delete a directory on a File Server.

The procedure is identical to the procedure FTP_Delete in 5.6 except that the command “Remove Directory” shall be used instead of the command “Delete”.

5.8 FTP_MakeDirectory

Use

This Management Procedure shall be used to create a directory on a File Server.

The procedure is identical to the procedure FTP_Delete in 5.6 except that the command “Make Directory” shall be used instead of the command “Delete”.

5.9 FTP_FileSize

Use

This Management Procedure shall be used to read the size of a specified file.

The procedure is identical to the procedure FTP_Delete in 5.6 except that the command “Get File Size” shall be used instead of the command “Delete”.

The File Size shall be returned with the Return Code “Command successful”.

5.10 FTP_EmptyDiskSpace

Use

This Management Procedure shall be used to read the size of the available free disk or memory space of the File Server.

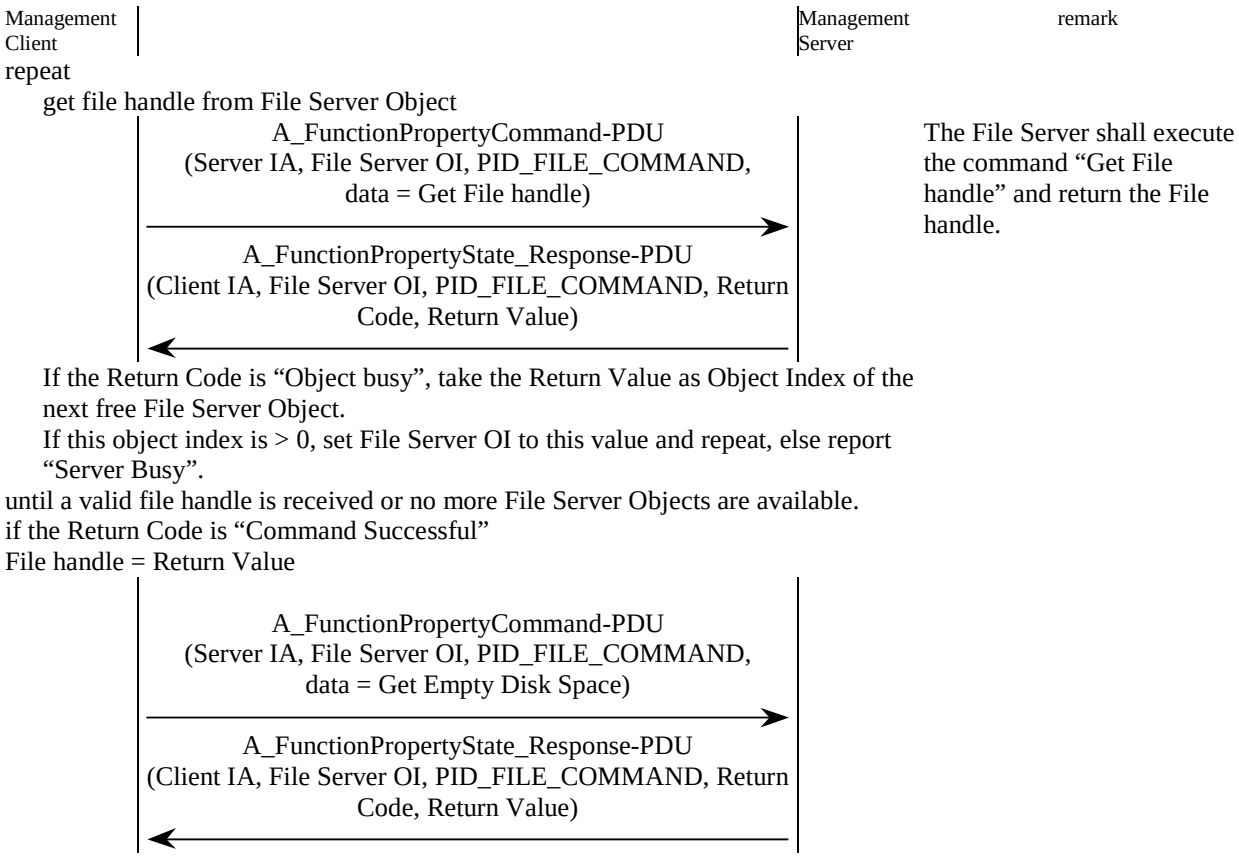
Used Application Layer messages for management

- A_FunctionPropertyCommand-PDU(destination_address, object_index, Property_id, data)
- A_FunctionPropertyState_Response-PDU(destination_address, object_index, Property_id, return_code, data)

Variables in FTP_EmptyDiskSpace

Server IA	Individual Address of the FTP Server
File Server OI	Object Index of the File Server Object in the Management Server.
Client IA	Individual Address of the FTP Client.
File handle	File handle retrieved from the FTP server

Sequence



Management Client		Management Server	remark

if the Return Code is “Command Successful”, the command was executed correctly
else an error shall be reported.
The File handle shall be released.

The Empty Disk Space is returned with the Return Code “Command successful”.

5.11 FTP_Abort

Use

This Management Procedure shall be used to stop any running transmission from a File Server. This is the only command that is sent without requesting a File handle. The valid File handle has been given to the client for the previous command (Retrieve File, List Directory or Get File) which has to be stopped by the Abort command.

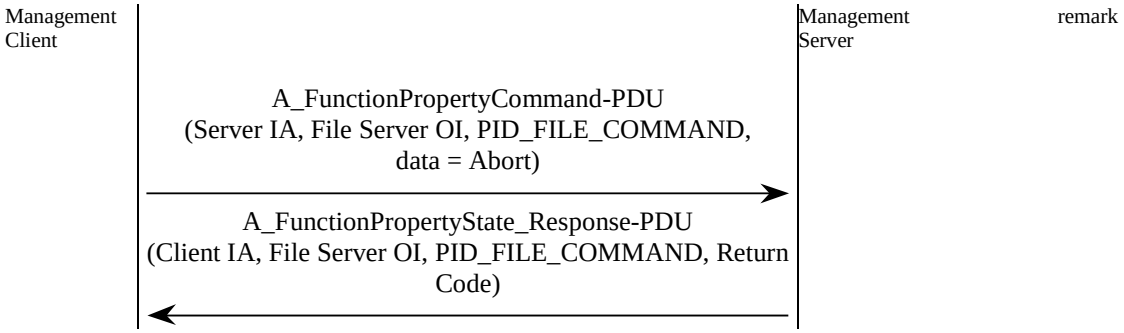
Used Application Layer messages for management

- A_FunctionPropertyCommand-PDU(destination_address, object_index, Property_id, data)
- A_FunctionPropertyState_Response-PDU(destination_address, object_index, Property_id, return_code)

Variables in FTP_Abort

Server IA	Individual Address of the FTP Server
File Server OI	Object Index of the File Server Object in the Management Server.
Client IA	Individual Address of the FTP Client.

Sequence



if the Return Code is “Command Successful”, the command was executed correctly
else an error shall be reported.

5.12 hTTP_GetFile

Use

This Management Procedure shall be used to read a file in hTTP mode from a File Server.

The procedure shall be identical to 5.2 “FTP_RetrieveFile”.

The Content-Type shall be returned with the Return Code “Command successful”.

5.13 hTTP_PostFile

Use

This Management Procedure shall be used to write a file in hTTP mode to a File Server.

The procedure shall be identical to 5.3 “FTP_StoreFile”.