Application Domain specific Standards

Logical Tag Extended

10

1

Summary:

This document specifies the Logical Tag Extended Mode (LTE-Mode), which is mainly designed to cover the specific needs of Easy Configuration for HVAC applications. LTE consists of a coexistent HVAC Easy Extension (HEE part) and a Standard System Interworking. LTE including HEE are standardised and part of the KNX Standard.

Main focus of this document is to describe the LTE installation procedures and the related specific protocol mechanisms for LTE.

Version 01.02.01 is a KNX Approved Standard.

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

Document updates

Version	Date	Modifications
0.1	1999.08.12	BKY, first draft
0.2	1999.12.01	BKY, improved combination of group-addressing and Interface Objects ⇒ new TPCI, AL-services, APDU-format new predefined usage of 16t bit group address range for zoning proposal for LTE “push-button” mechanisms for unidirectional device
0.3	2000.07.28	BKY, clauses 1 - 7.1 updated (excluding 7.1.1 and ff): the rest is outdated and partly obsolete
0.4	2000.09.12	BKY, completely revised document ⇒ release for CSG
0.5	2000.10.04	BKY, revised document, editorial and some minor corrections and amendments ⇒ release for CSG
0.6	2000.12.01	Templates for LTE-HEE Datapoint descriptions added Release for voting
0.7	2001.05.09	BKY, editorial: various amendments and usage of KNX standard document template All sections concerning unidirectional devices / push-button linking removed in clause 5.3, 7.2.3, 8.1, 11, 12 Interworking Model: amendments in clause 6.3; mainly clarification of LTE Read access in clause 6.3.3 Interworking Model: clarification of Property usage: new clause 6.5 Templates for LTE-HEE Datapoint descriptions updated: clause 6.5 LTE-HEE zone address tables updated: clause 0 LTE-HEE group address table format updated according to Supplement 5, Implementation Independent Resources: clause 7.3 Network management clause 8: some clarifications and details for existing mechanisms added Encoding of void binding links clarified: clause 9.2
1.0	2001.01.07	Preparation of the Approved Standard. There were no comments. Editorial enhancements, version update and spelling check only.
1.1 WD	2006.02.27	Editorial corrections & Integration of AN42
1.1 AS	2006.11.03	Supplement 5 "Implementation Independent Resources" clause 3 integrated Interworking Annex B.
1.1 AS	2006.11.07	Removed indication of DD8 in 8.3.
1.1 AS	2006.11.15	Correction of erroneous replacement of “Interface”, “interest”, “entered”, etc. by “Interworking” and of “Datapoint” by “Datapoints”.
	2006.12.20	Reformulation of clause 8.3 in view of integration of AN048.
	2007.11.14	**AN036 „Frame Type Parameter“** integrated.
	2007.11.16	**AN043 „LTE on RF“**integrated.
	2008.05.06	**AN067 „Unload IA for Easy Modes“**integrated.
1.1	2009.06.26	Update in view of publication in the KNX Specifications v2.0.
1.1.01	2009.10.29	Editorial update.
1.2.00	2012.03.07	**AN145 “Private IOs Property encoding in LTE-services”** integrated.
01.02.01	2013.10.28	Editorial updates for the publication of KNX Specifications 2.1.

References

[01] Part 3/2 “Communication Media”
[02] Chapter 3/2/5 “Communication Medium RF”
[03] Chapter 3/3/2 “Data Link Layer – General Requirements”
[04] Chapter 3/3/4 “Transport Layer”
[05] Chapter 3/3/7 “Application Layer”
[06] Chapter 3/5/1 “Resources”
[07] Chapter 3/5/2 “Management Procedures”
[08] Chapter 3/5/3 “Configuration Procedures”
[09] Chapter 3/7/2 “Datapoint Types”
[10] Volume 6 “Profiles”

Filename:	10_01 Logical Tag Extended v01.02.01 AS.docx
Version:	01.02.01
Status:	Approved Standard
Savedate:	2013.10.28
Number of pages:	114

Contents

1 Scope

The purpose of this document is to specify the LTE Easy Configuration mode (LTE-Mode):

- LTE configuration mechanisms and installer procedures,
- LTE network management mechanisms,
- LTE Interworking model and protocol extensions for runtime Interworking,
- Tool interface and KNX Standard system interface,
- LTE profiles and
- LTE testing.

The LTE-Mode is mainly designed to cover the specific needs of Easy Configuration for HVAC applications. LTE consists of a coexistent HVAC Easy Extension (HEE part) and a standard system interface in order to enable runtime Interworking with other S-Mode and E-Mode. LTE including HEE are standardised and part of the KNX Standard (Part 10/1 "Logical Tag Extended").

Specification of the HEE part is the focus of this document. Concerning the standard system interface, only specific aspects for LTE are described in this document.

Besides the specific protocol extensions for LTE-HEE runtime Interworking, all necessary KNX Standard protocol mechanisms are used and supported in LTE devices. These standard mechanisms are only referenced but not described in this document.

2 Summary

The Integration of Application Domains is a major goal of the KNX system. LTE is a solution designed mainly for HVAC applications in terms of special requirements for extended zoning information and large address space for data objects.

Besides LTE, other solutions for HVAC based on the standard system interface (S-Mode and E-Mode) may also be implemented.

In an LTE easy configuration system, a simple device configuration is possible for every installer without usage of a PC tool, without manufacturer databases, without knowledge of the object structures and of Group Addresses.

- Each device provides a means to enter zoning information (local on the device or remote). The zoning information is called ‘Tag’.
- Devices with same zoning information work together.
- Different zoning information may be entered in a given device. Each corresponding to a given, specified and predefined functional purpose.
- LTE mode enables a large number of groupings and Datapoints.
- This mode is applicable for several Subnetworks.

LTE-HEE Interworking aspects

- LTE-HEE runtime Interworking uses an extended message format which is not compatible to the S-Mode and E-Mode.
- LTE-HEE is coexisting to the standard system (S-Mode and E-Mode) on the same network (own standing system) using the same Physical Medium and L_Data_Extended frame on Data Link Layer.
- LTE-HEE is based on the usage of Interface Objects with Group Addressing (logical zoning) and allows much more flexibility concerning addressing space for the encoding of a large number of zones and Datapoints.
- LTE-HEE uses standard Datapoint Types and objects but may have specific Datapoint Types and objects for HEE runtime interworking (own object list).
- LTE products may bear the KNX Certification Mark if complying with both standard system interface and the HEE requirements. A part of the data objects (at least the mandatory objects acc. to the application specification) have to be implemented in HEE and standard system interface (runtime mechanisms, Datapoint Types) in order to allow runtime-Interworking with other devices in mixed installations.
- Products complying with the HEE part only cannot bear the KNX Certification Mark.
- LTE devices support all mandatory services for network management and tool support.

3 LTE main principles

3.1 System philosophy

LTE is based on a decentralised system philosophy with a static application binding mechanism.

Decentralised system configuration:

To set-up the binding links, each LTE device provides local zoning parameters (logical tags) which can be configured by the installer. Devices with the same zoning information work together.

Static application binding:

Within a given zone, devices may exchange various Datapoints according to a predefined application standard. Zoning information and Datapoint information (kind of data to be transmitted) are corresponding to a given, specified and predefined functional purpose (static definition).

Runtime Interworking:

LTE-HEE runtime-Interworking is mainly based on the producer - consumer principle using group addressing. LTE-HEE messages are „typed“, with semantic:
- Zoning information is mapped statically to Group Addresses
- Datapoints are addressed by Interface Object / Property mechanisms

3.2 Benefits of LTE-Mode for complex applications like HVAC

- HVAC installers are used to the concept of logical tags which represent an application-dependent zone like e.g. ‘boiler-no.’ or a building location like ‘apartment/room-no.’ etc.
  LTE supports structured grouping (nesting) with “Wildcard” addressing. Up to 3 group nesting levels are possible.
- LTE allows decentralised, independent configuration of each device. Even desktop configuration of individual devices is possible.
- Simple replacement or adding of new devices is possible without touching the rest of the system.
- Link visualisation: With the use of logical tags, the binding information is obvious on every device at any time.
- Link modification: links can be changed easily by modification of the logical tags on the corresponding devices. The rest of the system is untouched.
- Plug & Play functionality if no specific zoning is required: all logical tags have predefined default value at manufacture.
- Because of the static application model and binding mechanisms, unidirectional devices (receive/transmit only) are possible. This enables e.g. low cost sensor/actuator solutions on Radio Frequency or Power Line Carrier media.
- Instant operation after power-up (e.g. no “enrolment” mechanisms)
- No restrictions to one sub-network or medium ⇒ “easy routers” are supported
- Reliable solution with minimised software overhead: the direct mapping of logical tags to group-addresses on the network is predefined statically at design time (e.g. predefined fixed look-up tables). Any dynamic “negotiation” at run-time may create software-overhead and risk of failures.
- Sufficient and future proof address range for zoning and Datapoint addressing
- Same Datapoint addressing mechanisms (Interface Objects / Properties) for individual addressing and group addressing.

4 LTE-Installation procedures

4.1 Installation steps

1. Set domain address (on open media only) using standard mechanisms
2. Set individual address (device address and Subnetwork Address)
   - Subnetwork Address (SNA): every device has a medium dependent default value at manufacture. In case of multiple Subnetworks, the SNA can be set remotely by the corresponding router or by a tool. For LTE standard SNA setting mechanisms are used.
   - Device address: every device has a default device address (FFh) when unconfigured. Device address can be set locally (manual address management) or remotely assigned via the network by a tool, address server etc. For LTE standard device address setting and checking mechanisms are used.
3. Set functional device parameters if necessary
   e.g. application type in an „intelligent“ hot water temperature sensor: „flow temperature“ or „return temperature“
4. Set logical tag(s) to establish the binding links with other device(s):
   - a device has one or more logical tags to be set. The number of logical tags depends on the functional capabilities of the device, i.e. the number of functional links
   - at manufacture all logical tags have a default value which must be changed by the installer in zoned systems. Manual zone number management is done by the installer. In simple systems (not multiple zones of the same type) the devices can use the default value of the logical tags
     (⇒ "Plug and Play")
     - devices with the same zoning information work together in a distributed LTE system
     - logical tags are stored (non-volatile) in the device and can be read out or changed by the installer at any time
5. Configuration is complete.

Local configuration:

The installer can configure a device completely by setting all parameters with the local device user interface. No PC-tools, databases, network address servers etc. are necessary.

Remote configuration:

In an *ideal* situation, all parameters of all kind of devices could be configurable locally on the device itself (usage of dip-switches, code-wheels, keys & display etc.). In practice, simple devices like e.g. sensors don’t have a code-wheel, display etc. in today’s solutions. Therefore the local configuration of logical tags on theses devices could be problem.

Since all logical tags are “normal” configuration Datapoints, they can be accessed on the bus as standardised Interface Objects / Properties. Therefore remote configuration of logical tags is possible easily with standard mechanisms using a handheld unit or a controller based MMI etc.

For the remote configuration of network addresses (individual address, domain address etc.) the standard network management mechanisms and services are supported and configuration can be done remotely by standard tools (handheld unit or a controller based MMI etc.)

4.2 Modification of existing installations

4.2.1 Adding of new devices

An LTE installation can be extended easily – usually without touching the rest of the existing system. The normal installation steps as described in clause 4.1 must be followed.

4.2.2 Replacing of existing devices

In case of repair or functional upgrade, an existing device in an LTE installation can be exchanged easily.

No specific network mechanisms are required.

Steps:

1. Check all necessary parameter of the existing device (individual address, domain address, logical tags)
   - read parameters from the device itself (if possible, device not defective) or
   - consult plant diagram
   - consult commissioning report
2. Replace existing device by the new device and configure parameters as described in clause 4
3. Add new links (configuration of additional logical tags) if necessary in case of functional extensions

4.2.3 Removing of devices

An LTE device can be removed easily – usually without touching the rest of the system. If needed, corresponding functions in partner devices must be disabled (void links, see clause 9.1.)

No specific network mechanisms are required.

4.2.4 Link modification

An LTE device can be reconfigured easily. The corresponding logical tag(s) must be set according to the new needs.

5 Logical tags

To provide sufficiently generic binding mechanisms, different types of zones and corresponding logical tags are supported:

- Geographical tags: “Building Location” mainly used for room management in residential and commercial buildings
  ⇒ general mechanism for home automation and building/room management applications
- Application specific tags: used for application specific (functional) bindings not related with a building location
  ⇒ to be defined according to application needs
- Unassigned (peripheral) tags: bindings without functional purpose;
  ⇒ general mechanism; mainly for binding of general purpose sensors and actuators

The usage of these classes of LTE logical tags is defined in the Application Specifications on the level of Functional Blocks and Datapoints. LTE logical tags are configurable parameters (i.e. properties) of Functional Blocks.

Rule

Logical tags are only used for zoning information (Link Layer information) but not for Datapoint identification.

5.1 Geographical tags

In a HBES (Home & Building Electronic System) many functional groupings can be associated with locations in the building, like an apartment/floor or a room or a sub-zone in a room. These building locations are mapped to logical tags like an ‘Apartment-No.’ / ‘Floor-No.’ and ‘Room-No.’ and ‘Subzone-No.’

This binding mechanism is application independent and could also be used for other applications than HVAC.

5.1.1 Addressing requirements for geographical tags in residential installations

In residential installations the addressing of a large number of apartments is required (e.g. for large apartment buildings or district heating system with a large number of single family homes). The max number of rooms per apartment is uncritical.

The geographical tag is hierarchically structured in 3 building location levels:

Apartment-No.Room-No.Subzone-No

Required number of zones:

- 126 Apartments
- 15 Rooms per Apartment
- 15 Subzones per Room
- 15 Subzones per Apartment

5.1.2 Addressing requirements for geographical tags in commercial installations

In commercial “easy” installations the addressing of a small number of floors is required. But in contrast to residential applications, a large number of rooms per floor are required.

The geographical tag is hierarchically structured in 3 building location levels:

Floor-No.Room-No.Subzone-No

Required number of zones:

- 15 Floors
- 63 Rooms per Floor
- 15 Subzones per Room
- 15 Subzones per Floor

5.1.3 Combined addressing scheme of residential/commercial geographical tags

LTE geographical tags in residential and commercial applications are both hierarchically structured in 3 building location levels. The only difference is the **naming** of the 1st level: apartment / floor. In order to allow products to be used in residential and commercial installations a common building location addressing scheme is defined.

The combination of the requirements for residential and commercial building location tags shows a need for a large number of apartments in residential installations and a large number of rooms in commercial applications. The number of subzones per room is identical. This leads to the following combined solution:

Building Location Level		Number of Zones
1st	Apartment Floor	126
2nd	Room	63
3rd	Subzone	15

Encoding & transmission of this structured building location is the same for residential and commercial applications. Only the interpretation / naming of the 1st level for the user may be differentiated case by case.

5.1.4 Wildcard addressing

For each level of building location “Wildcard-Addressing” is supported:

1st level	2nd level	3rd level	
A/F	*	*	all in the same Apartment/Floor (Room „Wildcard“ and Subzone „Wildcard“)
A/F	R	*	all in the same Apartment/Floor.Room (Subzone „Wildcard“)
A/F	*	S	all Subzones with the same Subzone-No in the same Apartment/Floor (Room „Wildcard“)
*	R	*	all Rooms in the installation with the same Room-No. (Apartment/Floor “Wildcard” and Subzone „Wildcard“)
...	...	...	etc.
*	*	*	„Wildcard“ all in the building / installation

This is a very powerful mechanism used by the **Sender** of a message in order to access all Functional Blocks assigned to the same building level(s): e.g. to all devices in the same apartment or the same room etc.

Application examples:

- An apartment scheduler sends the HVAC mode to all individual room control systems in the apartment with the zoning information A.*.*
- Non HVAC example: central light switch information could be sent per room (A.R.*) or per apartment (A.*.*)
- Non HVAC example: a management station could send one command to close all blinds on the south side (e.g. = subzone S in a given installation) of the building (*.*.S)

5.1.5 Configuration levels and “Sniffer” functionality

Depending on the functionality of a device, only a part of the 3 building location levels must be configured as a logical tag. The other levels are “don’t care”.

Application examples:

- For an apartment heating controller (no individual room control) it is sufficient to configure the Apartment-No. The other tags Room-No and Subzone-No are not configurable because they are not needed explicitly in this application
- There is only one room temperature in a room. Therefore the tag Subzone-No is not configurable on a room temperature sensor.

Building location "Sniffer" functionality in the Receiver:

Messages containing explicit building location information are received (at least at the level of zoning, but not necessarily at Datapoint level) by Functional Blocks / devices belonging to a higher building level structure.

Application-examples

- A message from an individual room controller IRC containing the building location A.R.* shall also be received in an apartment MMI which only is configured with the tag Apartment-No.
  ⇒ The MMI is a "Sniffer" on all Room-No in the same Apartment
- An individual room controller which is configured with the tag Apartment-No.Room-No will receive messages also containing explicit subzone information A.R.S
  ⇒ The individual room controller is a "Sniffer" on all Subzone-No in the same Apartment.Room
- a building management station (geographical tag = *.*.*) will receive all messages containing building location zoning information

5.1.6 Application examples

Remark: LTE mechanisms are introduced for HVAC only. The following explanatory examples show some theoretical possibilities with other applications.

Producer / Data	Consumer	Binding link
- IRC heating controller: no zoning for valves per room - Data: valve position setpoint	Valves	A.R.*
- IRC heating controller: 2 or more zones per room (e.g. combination of radiator and floor heating in the same room) - Data: valve position setpoint	Valves	A.R.S
- Presence sensor (room level) - Data: presence status	IRC heating system Apartment security system Room blinds controller	A.R.* A.R.* A.R.*
- Window status sensor - Data: window status	IRC heating system: for IRC it is not important, which window is open in the room Apartment security system	A.R.* A.R.* or A.R.S
- Room temperature sensor - Data: room temperature	IRC heating controller Apartment heating controller (reference room) MMI room MMI apartment	A.R.* A.* A.R.* A.*
- Apartment Manager - Data: apartment-mode (lifestyle, occupancy)	IRC heating system Apartment heating controller Apartment security system Room blinds controller	A.* A.* A.* A.*
etc.		

Example: IRC heating system including blinds control

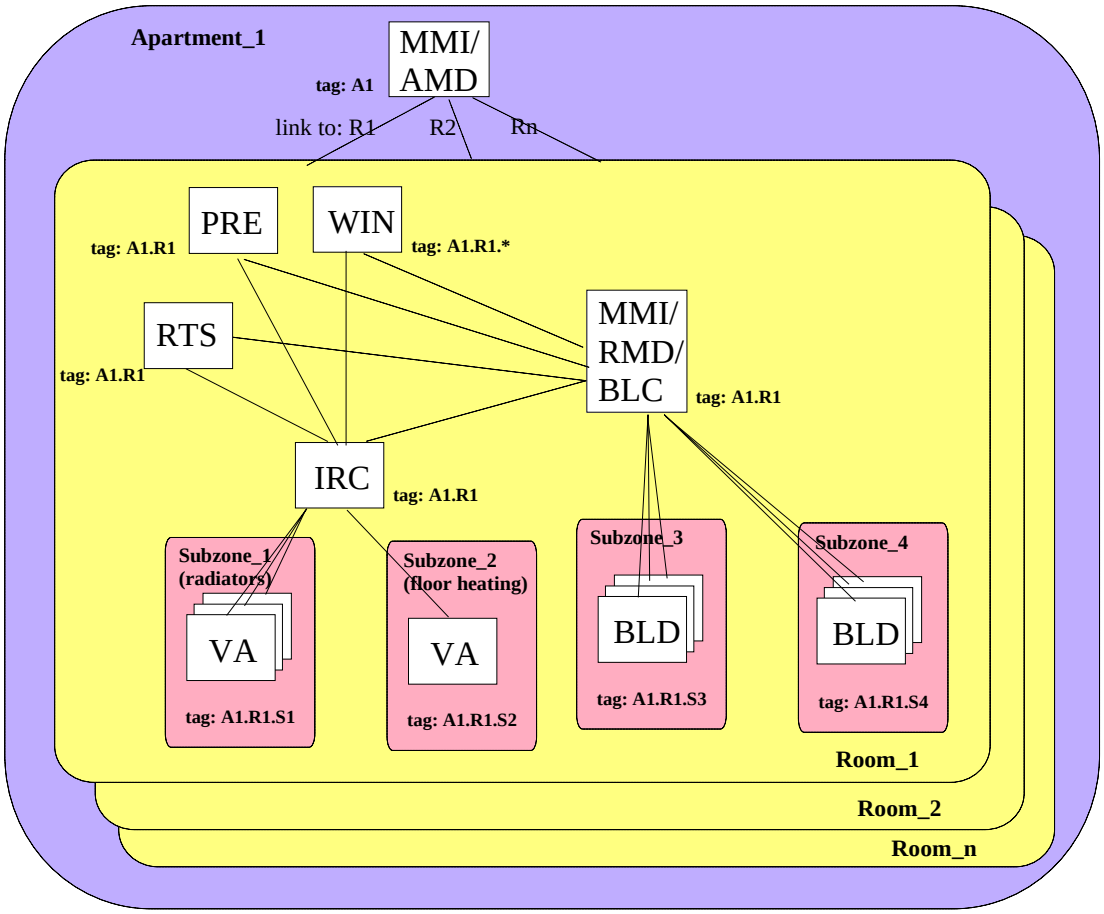


Figure 1 - IRC heating system including blinds control

MMI/AMD:	*Man machine interface/Apartment management device*		
MMI/RMD:	*Man machine interface/Room management device*		
IRC:	*Individual room controller*	*VA:*	*Valve*
BLC:	*Blinds controller*	*BLD:*	*Blinds drive*
RTS:	*Room temperature sensor*	*PRE:*	*Presence sensor*
WIN:	*Window status sensor*		

The example above shows one room of an apartment with:

- an individual room control system (heating) with one IRC controller and valves in 2 subzones:
  n valves for the radiator circuit and one valve for the floor heating circuit
- one blinds control system which is combined with the IRC system
  The blinds in the room can be controlled individually in 2 subzones
- a room temperature sensor for the heating system
- a presence sensor and a window status sensor for the heating system
- a MMI/Room management device: centralised user-interface plus room management functionality, including blinds control function
- a MMI/Apartment management device allows to control central functions of the apartment (schedule, lifestyle/ house mode, parameter settings, diagnostics) etc.

5.2 Application specific tags

For applications like HVAC, additional application-specific binding groups and the corresponding logical tags are required besides geographical tags.

E.g. logical tags for:

- Hot Water production segments and heat producers (boilers) ⇒ ProdSegmH-No.Producer-No
- Cold Water production segments and producers (chiller) ⇒ ProdSegmC-No.Producer-No
- Distribution segments for hot water ⇒ DistrSegmH-No
- Distribution segments for cold water ⇒ DistrSegmC-No
- Distribution segments for ventilation ⇒ DistrSegmV-No
- Domestic hot water zones ⇒ DHWZone-No
- Outside sensor zones ⇒ OutsideSensorZone-No
- HVAC calendar zones ⇒ HVACCalendarZone-No
- etc.

These application-specific groups are independent of a building location e.g. A.R.S

The number of these application-specific groups is rather small compared to the required number of building location groups (in terms of Group Address space).

Some of these application specific tags may also support nested zoning as described for geographical tags in clause 5.1.4 ⇒ support of wildcard and “Sniffer” feature

The application-specific groups are defined by the Application Interworking Specifications. For HVAC Hot Water Heating (HWH) the HWH-specific groups are defined. See explanatory examples in the Appendix.

5.3 Unassigned (peripheral) tags

Some sensor/actuator functional groupings can’t be associated directly with locations in the building or application-specific groups. This is particularly the case for multipurpose sensors/actuators with a 1:1 link to another device (e.g. a pump connected to a heating controller). These devices could be used for different purposes and therefore the functional binding using a dedicated „location“-group like ‘Apartment’, ‘Room’ etc. is not appropriate.

Therefore the concept of unassigned tags (general purpose binding links) is introduced. These links are also configured by a logical tag.

Unassigned peripheral tags are not structured and do not support nested zones, i.e. no wildcard or “Sniffer” features.

6 LTE Interworking model

This section contains LTE-HEE specific parts only which are an addition to the standard KNX Interworking Model.

6.1 Introduction

Modular application modelling by usage of Functional Blocks is a common concept for all KNX applications – independent of the communication model.

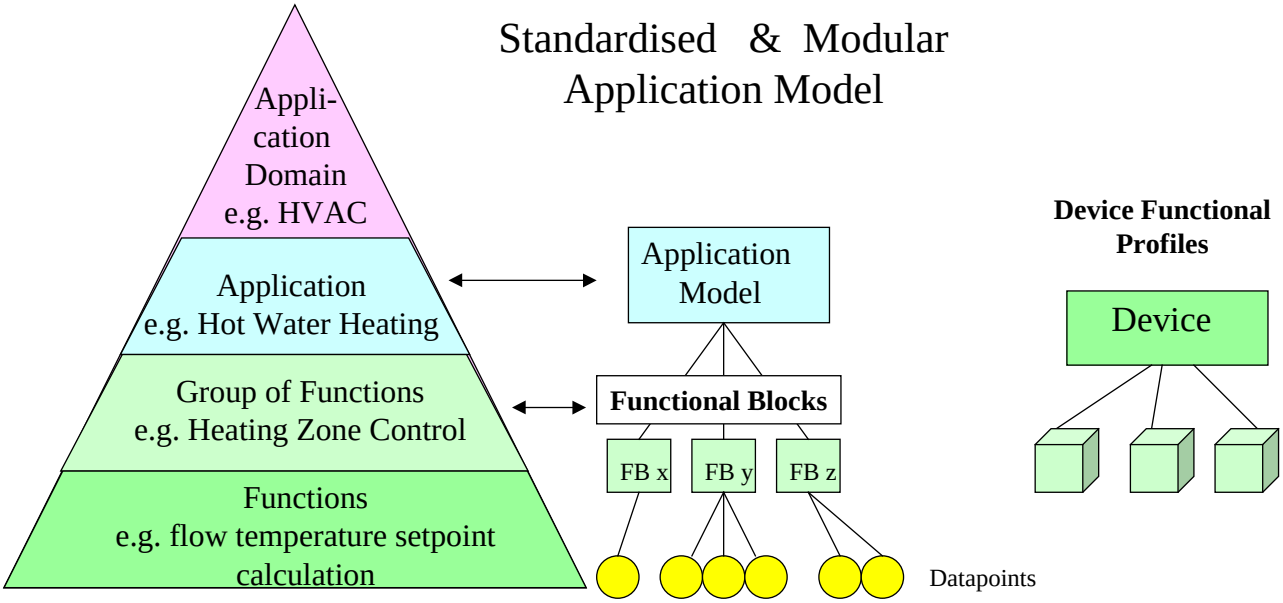


Figure 2 - Top down application modelling using Functional Blocks

In **S.E.A implementations** the concept of Functional Blocks is only used for logical application modelling but it is no more visible at message-level used for runtime Interworking.

In the **LTE-HEE implementation**, the concept of Functional Blocks is maintained in the message structure. I.e. the Functional Block is part of the data addressing mechanism used for:

- device oriented data access: using individual device addressing
- multicast communication: using group addressing
- in both device oriented and group communication services, Datapoints are accessed via Interface Object (=Functional Block) / Property mechanisms

6.2 Overview: mechanisms for LTE-HEE runtime Interworking

Two main classes of data objects used for runtime Interworking can be identified in a complex HBES system:

- **Management objects**: data objects exchanged between a device (server) and a user interface (client), used for configuration/parameterisation and diagnostics.
- **Process objects**: data objects for process- / runtime-Interworking in distributed applications. These signals normally are caused by an automated physical process or by user interactions.

6.2.1 Client/server based management objects for interactive operations (MMI)

If a control application needs to read information from a remote object on-demand, or needs to write information to specific object and obtain confirmation of the success of the write operation, the device-oriented Client/Server paradigm using **peer to peer individual addressing** will normally be used.

These messaging requirements occur under the following situations:

- Installation/upgrade of a system using a configuration tool, e.g.,
  - The installer wishes to write configuration information/parameters to specific devices.
  - During a system upgrade, the installer wishes to understand the existing system by reading configuration information / parameters from specific devices.
- Fault diagnosis with a service tool, e.g.,
  - The technician wishes to read device configuration information, or operational status information from specific devices.
- During system run-time if a user of a MMI requires an immediate display of up-to-date information (for specific device data not accessible by group addressing).

The client needs to know the individual address of the server to be accessed. I.e. the client must maintain a local device directory of its partner device(s). Data is exchanged using confirmed connectionless AL Read/Write services (A_PropertyValue_Read and A_PropertyValue_Write services)

In LTE devices, these management objects are implemented as 'Properties' of standard 'Interface Objects' with individual device addressing and are not discussed further in this document.

6.2.2 LTE process objects: producer/consumer based runtime Interworking

The vast majority of information transfer within a distributed HBES system at run-time occurs periodically (using "heartbeat" period), or when a data value changes (COV, change of value mechanism).

The Producer/Consumer or Shared Variable paradigm using **multicast/group-addressing** is ideal for use in these situations. The consumers of data wait until the producers of data decide that new data is available, and must be sent.
The use of the Client/Server paradigm for runtime operation would normally be inefficient in terms of network bandwidth utilisation since a client must send a request in order to receive its data. Also, multiple clients may requests the same data resulting in multiple copies of the data being sent to individual clients. The Producer/Consumer paradigm provides an elegant mechanism to distribute process data to those who require it by using multicast (group-) addressing. In this way, multiple consumers are able to utilise the same data from a single message.

With Client/Sever method, the client would need to know the server's device address, and device registration (enrolment) is typically required.

6.2.3 Functional block data interface

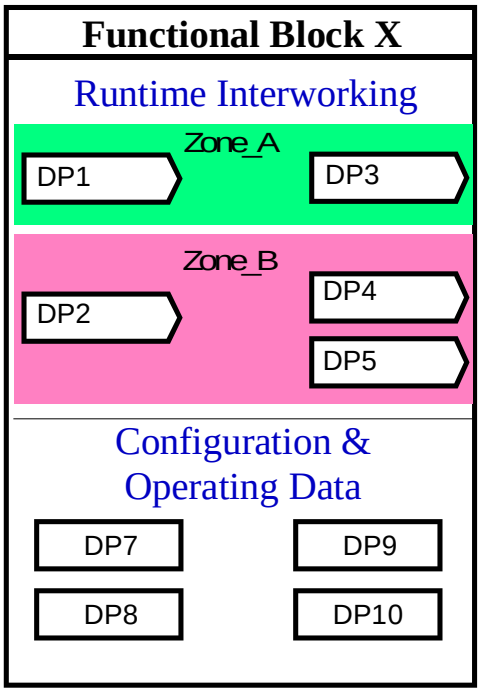


Figure 3 - Functional block data interface

The figure shows a Functional Block with:

- LTE process objects: input Datapoints (left side) and output Datapoints (right side) for runtime-Interworking and their corresponding zoning information. These Datapoints are transmitted using group communication

  Some of these Datapoints are also accessible via standard system mechanisms (S-mode interface) This standard data interface is not shown in the figure above.

- Management objects: configuration and operating data used on an MMI or management station: to be implemented as Properties of Interface Objects and accessed via client/server mode and individual addressing.

For each Datapoint the following characteristics are defined in the Application Standard:

- Datapoint address
- Datapoint type
- upper &lower limits, default values
- data distribution and data access mechanisms
- actions on data
- access rights

For LTE-HEE process objects the corresponding zoning information is also defined in the Application Standard.

6.3 Data exchange mechanisms using LTE-HEE group communication

The LTE-HEE system offers 4 different data exchange mechanism using group addressing. The proper usage of each mechanism is application-dependent and has to be decided and specified case by case for each Datapoint.

6.3.1 InfoReport mechanism

Information Report service (abbreviation InfoReport) is used to inform interested consumers in the same zone about an updated data value or about an event. A Functional Block shall transmit Datapoint information using InfoReport service and group addressing if:

- The message is sent spontaneously by the Functional Block (COV or heartbeat)
- The Functional Block producing the message does not know or care about the receivers of the message. I.e. source “address information” of the message is important but it does not matter to which Functional Blocks it is sent
- The producer does not know the number of consumers: the service is unconfirmed and loss of data due to communication failure may occur
  ⇒ increased reliability with heartbeat repetition besides COV mechanism
- The Functional Blocks consuming the message are “free” to store the data in the database or use it to trigger a function or use it temporary for some calculation or just ignore it etc.
- Data may contain an additional Status Field besides the main value information (e.g. ‘Overridden’, ‘OutOfService’ etc.)

The Functional Blocks consuming the message:

- Are in the same zone as the producer
- Must “know” the originator (type of Functional Block and Datapoint) of the message
- May have a permanent image of the received data – but local read-back capability (using individual addressing) is optional.

Information Report service is often used in 1:N or 1:1 relations where the producer does not know the consumers of the data. E.g.

- a sensor Functional Block distributes its sensor value to “whom it may concern”

Handling of “transient” information (event notification)

Usage of InfoReport service is ideal to provide “event notification” information. For this kind of “transient” information slow heartbeat repetition of the signal does not make sense. Therefore the message will usually be transmitted only once. If higher reliability of transmission is required by the application, the same message may be transmitted consecutively several times.

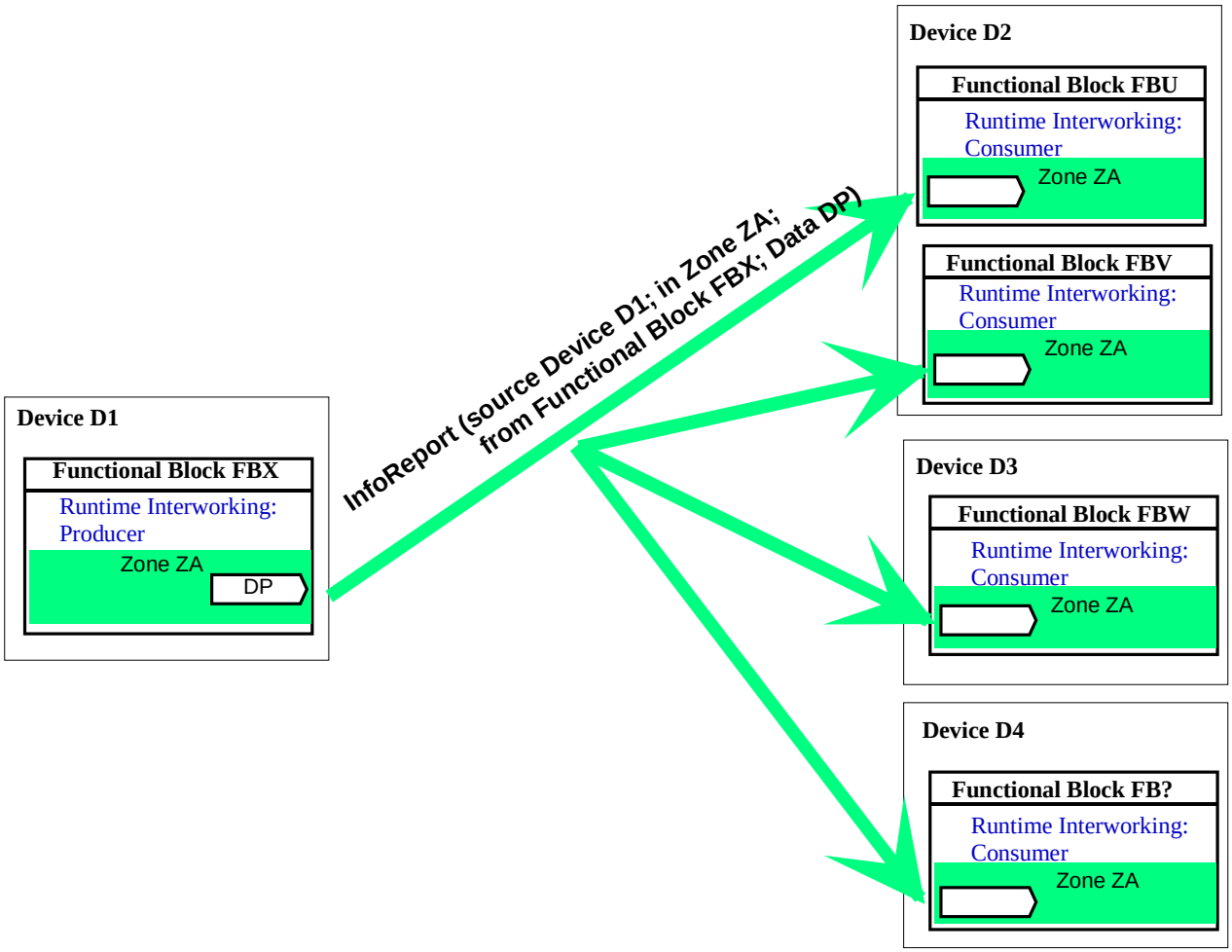


Figure 4 - InfoReport mechanism

LTE InfoReport Outputs are generally readable using LTE Read service because the sending Functional Block is the owner of the Datapoint (server), see clause 6.3.3

On the other hand InfoReport Inputs can not be read-back using LTE Read service because the receiver is only the client of the Datapoint! But instead, a local copy of the Input can be implemented as a Property of the receiving Functional Block. This Property would be accessible using individual addressing. This feature shall be defined in the Application Specification.

6.3.2 Write mechanism

Write service is used to change data in the database of a group of Functional Blocks in the same zone. This LTE-HEE mechanism corresponds to the S.E.A shared variable principle. A Functional Block shall transmit Datapoint information using Write service and group addressing if:

- The message is sent spontaneously by the Functional Block (COV or heartbeat)
- The Functional Block sending the message knows the type of the Functional Block(s) which shall store the written data in their database. I.e. destination “address information” of the message is important and contains the Functional Block type and Datapoint address information of the receivers. But it does not matter from which Functional Block the message was sent. Only one type of Functional Block and Datapoint may be accessed by one write request.
- The sender does not know the number of receivers: the service is unconfirmed and loss of data due to communication failure may occur ⇒ increased reliability with heartbeat repetition
- Data may contain an additional Command field besides the main value information (e.g. ‘Override’, ‘Set OutOfService’ etc.)

The Functional Blocks receiving the message:

- are in the same zone as the sender
- are explicitly addressed by their own Functional Block type and Datapoint
- don’t “know” the originator (type of Functional Block) of the message
- must keep a permanent image of the data
- local copy of the data must be readable (using individual addressing)

Write service is often used in 1:1 or 1:N or N:1 relations where the sender wants to control the receiver(s) of the data. E.g.

- a control Functional Block writes a setpoint to the corresponding actuator(s)

Handling of “transient” information (trigger)

Usage of Write service is ideal to send a “trigger” command to a defined group of Functional Blocks of the same type. For this kind of “transient” information slow heartbeat repetition of the trigger signal does not make sense. Therefore the message will usually be transmitted only once. If higher reliability of transmission is required by the application, the same message may be transmitted consecutively several times.

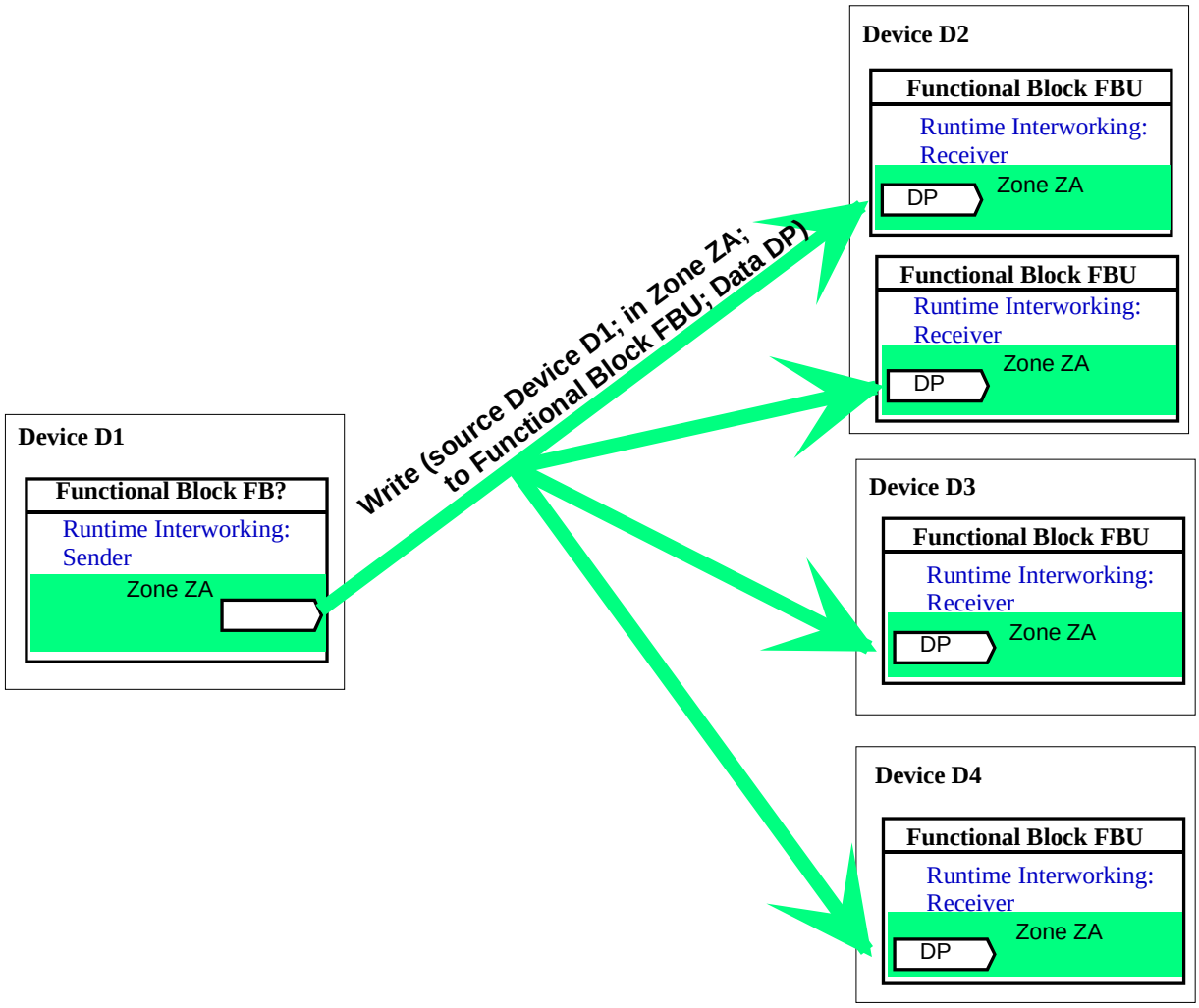


Figure 5 - Write mechanism

In the figure above two Functional Blocks of the same type which belong to the same zone are located in the same device D2. Both Functional Blocks receive the Write message and keep a copy of the data in their database.

LTE Write Outputs are generally NOT readable using LTE Read service because the sending Functional Block is not the “owner” of the Datapoint (i.e. the client).

On the other hand, from a communication point of view, LTE Write Inputs could theoretically have read-back capability using LTE Read service because the receivers are in principle the Datapoint “owners” (i.e. the servers), see clause 6.3.3.

But in practice there are only few applications where reading-back of LTE Write inputs using multicast LTE Read service is meaningful. An LTE Read request could result in multiple responses.

Conclusions

- LTE Read access to LTE Write Inputs is not a general mechanism and is per default not supported.
- If LTE Write Inputs have LTE Read-back capability, this feature shall be defined in the Application Specification.
- The value of LTE Write Inputs is always accessible using Property Services and individual addressing.

6.3.3 Read/Response mechanism

Read/Response service is used to access data in the database(s) of a group of Functional Blocks in the same zone:

- A Functional Block requesting data by Read service is the client.
- The Functional Block(s) generating the Response(s) is/are the server(s).
- Client and server(s) Functional Blocks must belong to the same zone
- **Beware**: Read request by the client may lead to multiple responses and may overload the bus ⇒ usage of this mechanism to be decided carefully in the Application Specifications
- A Datapoint in a server may be read by multiple independent clients.

Read/Response mechanism is often used in 1:1 or 1:N relations where a client wants to poll data from one or multiple servers at a certain time.

- it should be used only if the client application needs up-to-date information **on demand,** e.g.
  – in an MMI due to user request
  – in order to get actual values after power up of the client
  – to read out parameters via group addressing once for application start-up etc.
- it should **not be used** for normal process data exchange based on COV or heartbeat mechanisms because InfoReport or Write services are more efficient (only one message instead of two)

Client side

- The read request is sent spontaneously by the Functional Block (event-driven or periodical polling)
- The client Functional Block sending the read request knows the type of the Functional Block and Datapoint which shall be accessed. I.e. destination “address information” of the message is important and contains the Functional Block type and Datapoint address information of the receivers. But it does not matter from which client Functional Block the message was sent. Only one type of Functional Block and Datapoint may be accessed by one read request.
- The client does not know the number of servers and responses: the Read service using group communication is therefore unconfirmed on the level of the Application Layer and loss of responses due to communication failure may occur. In the Application Process an application dependent Timeout for Responses may be implemented
- Read request contains no data

Server side

- After reception of the Read message (Read.ind) the server will read the Datapoint value out of its database and create a Response message within a “reasonable” time (usually some hundred milliseconds)
- The Response message is sent via group addressing (in the same zone as the received Read message) and contains the (source) Functional Block and Datapoint address of the server and the requested Datapoint information.
- Data may contain an additional Status Field besides the main value information (e.g. ‘Overridden’, ‘OutOfService’ etc.)
- the Response message structure is same as used in InfoReport Service

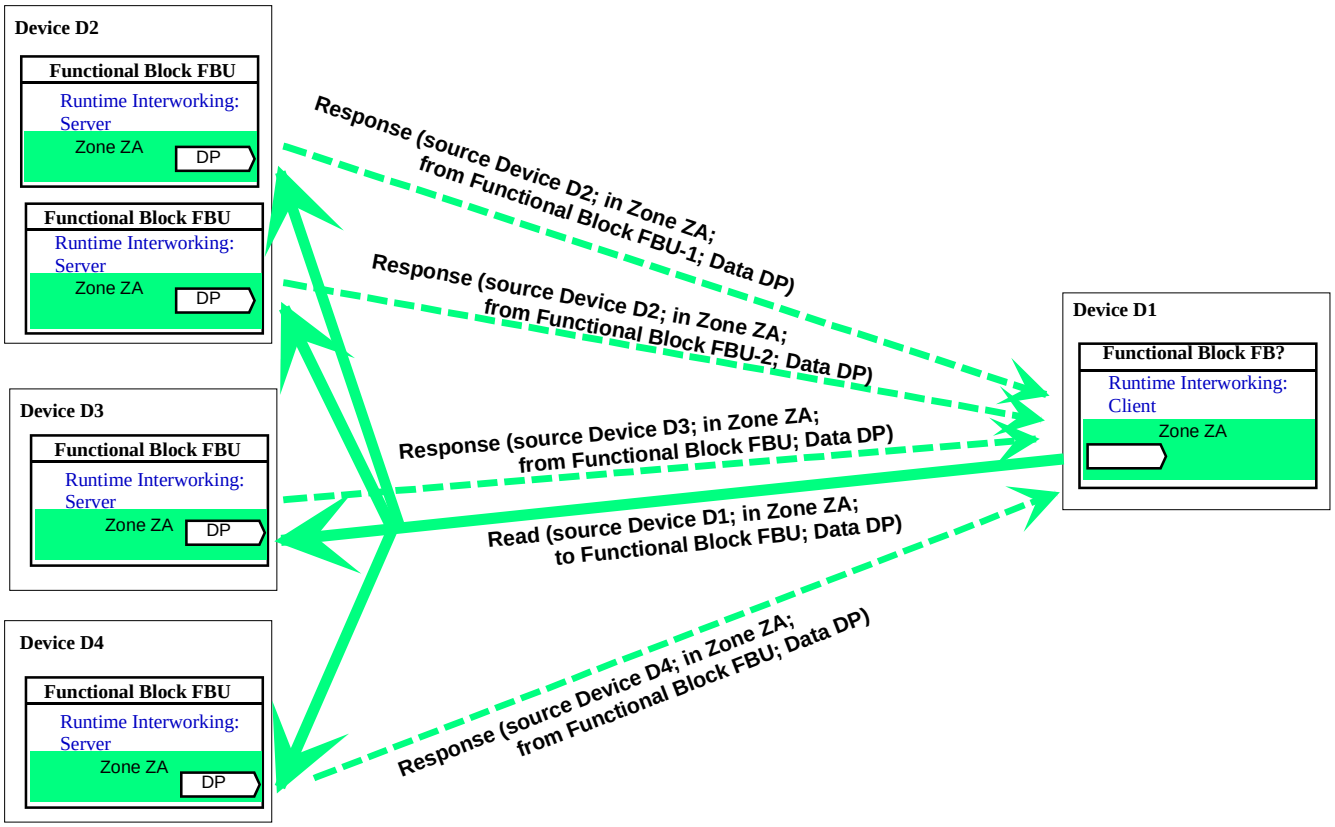


Figure 6 - Read/Response mechanism

In the figure above two Functional Blocks of the same type which belong to the same zone are located in the same device D2. Both Functional Blocks receive the Read message and will independently generate a Response containing the source Functional Block / Datapoint address and the corresponding data. In this example, multiple instances of the same Functional Block type in one device must be identified.

LTE Read access to Output Datapoints distributed by InfoReport Service:

- Datapoints (outputs) of Functional Blocks which are distributed using InfoReport service are in principle always readable in addition, and the consumers of the InfoReport will also accept the Response message and handle it like an InfoReport. See also clause 6.3.1.

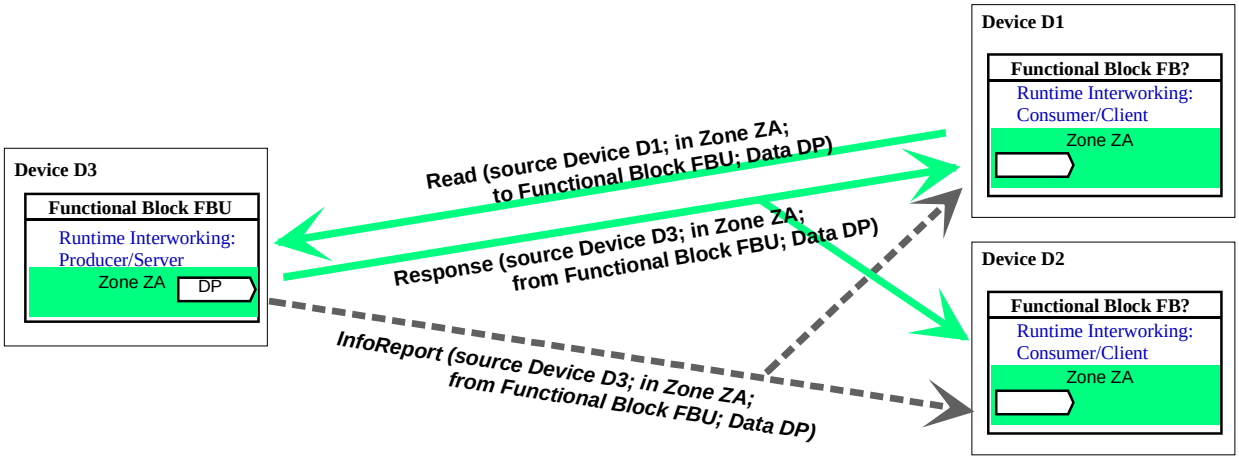


Figure 7 - Combined LTE InfoReport and Read/Response mechanism

Influence of a Read/Response Datapoint access on COV and heartbeat mechanism:

- If an output Datapoint is transmitted with InfoReport service using COV and/or heartbeat mechanism, a Read access to this output Datapoint shall not have any influence on the COV or heartbeat in the sender.
- I.e. the heartbeat period must not be re-triggered by the Read access
- I.e. the COV criterion is calculated based on the Datapoint value of the last InfoReport transmission.

Read access to “transient” output Datapoint

As described in clause 6.3.1 “transient” output information like an event notification is sent using InfoReport service. These output Datapoints are in principle also readable (see section above) but for this kind of information Read access does not make sense and the Datapoint value will be e.g. ‘False’ (no trigger) etc.

LTE Read access to Datapoints distributed by Write service

- **Outputs** of Functional Blocks which are sent using LTE Write service are generally NOT readable using LTE Read Service because the sender is the client of the Datapoint.
  See clause 6.3.2.
  If needed (e.g. as a diagnostic value), a copy of the output value may be implemented as a local Property in the sender. This Property could be Read from the sender using individual addressing.

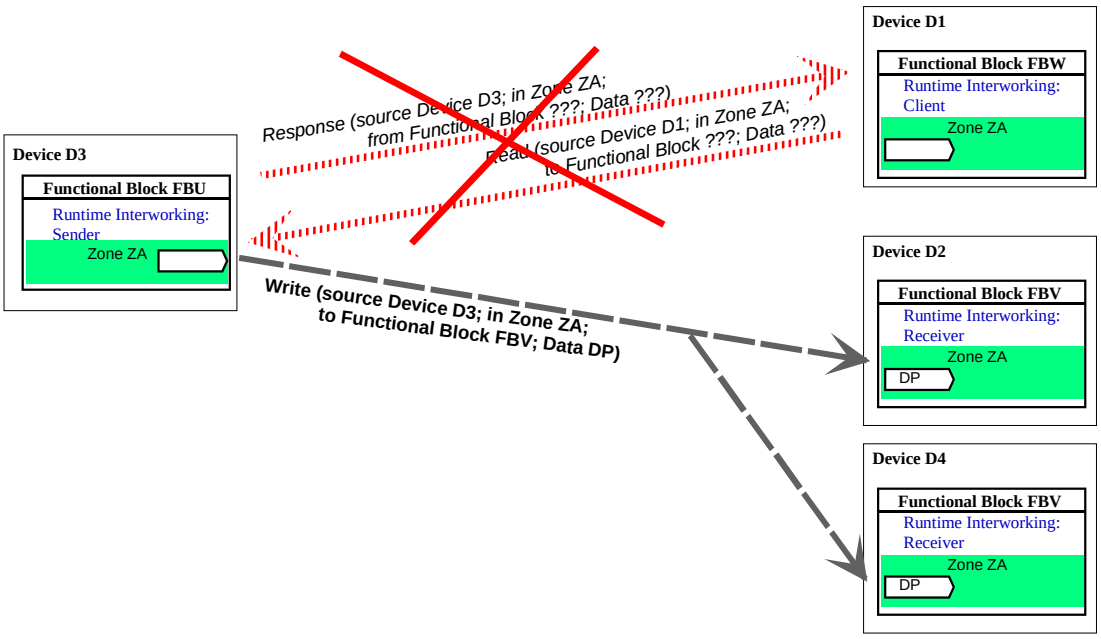


Figure 8 - No LTE Read access to LTE Write Outputs

- As described in clause 6.3.2, LTE Write **Inputs** do per default NOT support LTE Read-back capability although the receiving Functional Block is the Datapoint server.
  LTE Read access to LTE Write Inputs is an optional (and rarely used) feature to be defined by the Application Specification. If supported, the behaviour shall be as follows:

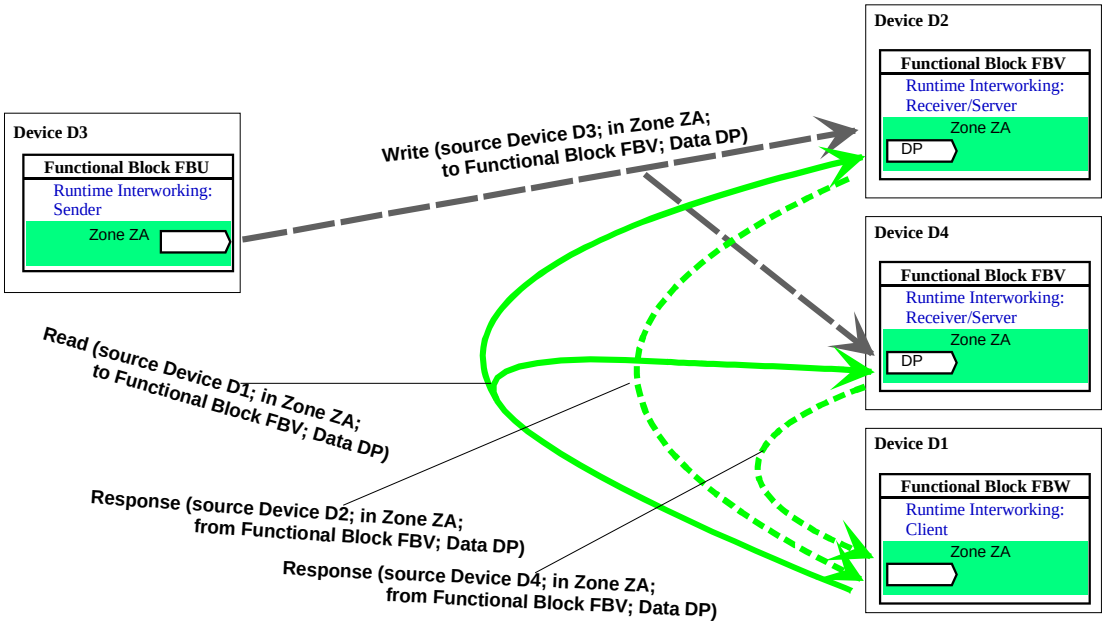


Figure 9 - Optional LTE Read access to LTE Write Inputs

6.3.4 Usage of LTE-HEE group communication services

The choice between Write, InfoReport or Read/Response mechanism has to be made for each LTE-HEE runtime data object. Decision is made in the Application Interworking Groups depending on the application needs respecting the specific features of each mechanism.

FB relationship	Usage of mechanisms
1:1	In 1:1 relationship the Functional Block knows its partner - **InfoReport**: if the consumer is not obliged to react and store the data - **Write**: if the receiver has to react and store the message - **Read/Response**: data exchange only on demand to reduce traffic
1:N	In 1:N relationship data from one Functional Block is sent to N Functional Blocks of the same type or different types. - **InfoReport**: if data shall be distributed to different types of consumers in parallel or potential consumers are not known, e.g. used in sensors - **Write**: if data shall be sent to a dedicated group of Functional Blocks of the same type and the data shall be stored in the receivers, e.g. actuators. If N different types of Functional Blocks shall be addressed the message must be written N times. - **Read/Response**: not for normal process data exchange May be used by any of the N consumers to get an data update from the one producer instead of waiting for the next InfoReport message
N:1	In N:1 relationship data from N Functional Blocks of the same type or different types is sent to one Functional Block. - **InfoReport:** to be used if the source Functional Block of the message is important to know in the one receiver - **Write**: N Functional Blocks of the same type or different types want to write a Datapoint in one known Functional Block. Alternate writing by Functional Block A or B to Functional Block C does change the database in Functional Block C - but A and B do not synchronise each other - **Read/Response**: not for normal process data exchange. May be used by the one consumer to get an data update from the N producers instead of waiting for the next InfoReport messages
N:M	In N:M relationship data from N Functional Blocks of the same type or different types is sent to M Functional Blocks of the same type or different types - Mix of 1:N and N:1 $\Rightarrow$ needs careful study of the application needs. Probably **InfoReport** is the preferred solution since the senders do not need to know the potential consumers of the data.

6.3.5 Usage of zone Wildcard and “Sniffer” mechanism in LTE services

As described in clause 5.1.4 structured zoning information may contain wildcards, i.e. the corresponding zoning level is interpreted as “don’t care” in the receiver of the message.

LTE message zoning information may contain **Wildcard** information in the following services:

- **InfoReport:**
  e.g. a Room Temperature Sensor reports its temperature value with the zoning information A.R.* (subzone within the room is a Wildcard)

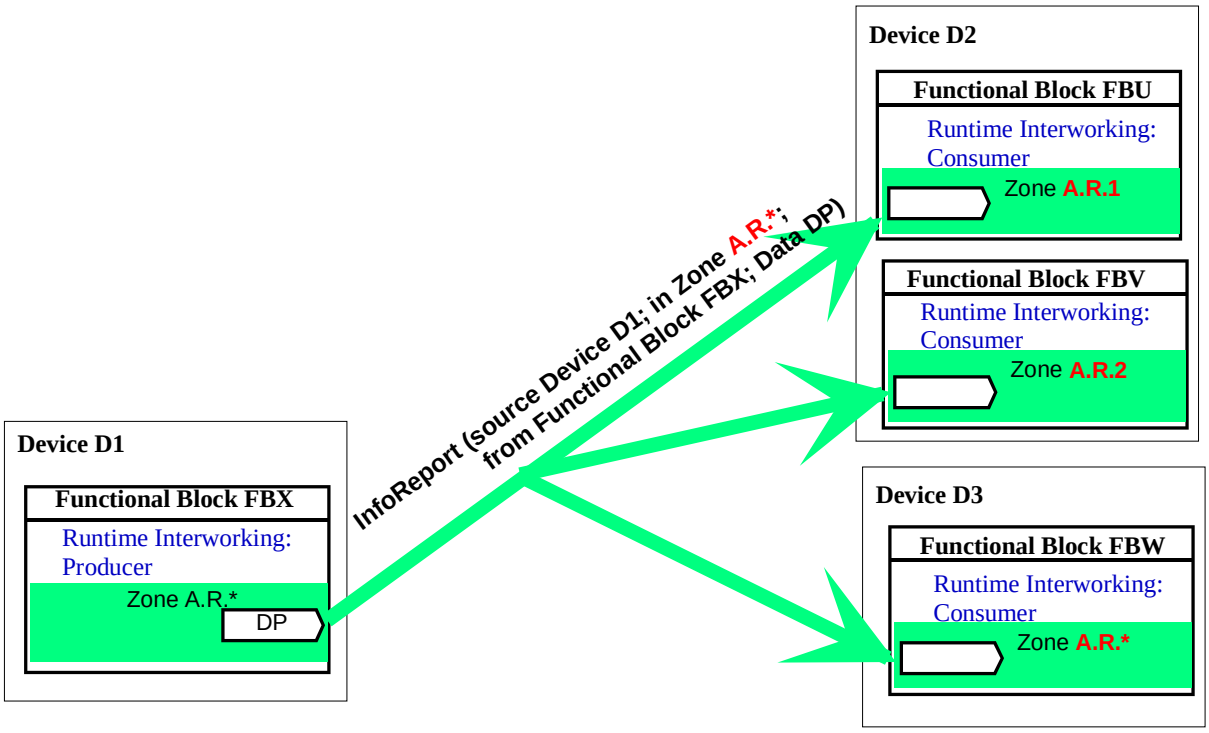


Figure 10 - Example: InfoReport using Wildcard zoning information

- **Write:**
  e.g. an Apartment Manager writes a control information or setpoint to all individual room control systems in the apartment using zoning information A.*.*

- **Read/ Response:**
  – in the **Read** request from the client, the usage of zoning Wildcard is a powerful mechanism to get information from a group of zones. But this may result in a lot of Response messages and overload the bus
  ⇒ to be used with care
  e.g. an MMI wants to read actual room temperature values from all rooms in an apartment using zoning information A.*.* in the read request

  – In the **Response(s)** of the server(s) the zoning information shall be as explicit as possible
  i.e. for the read example above, the room temperature sensor shall respond using explicit zoning information A.R.* and not A.*.* since the detailed zoning information is needed in the MMI

  This is a first example of zoning **“Sniffer”** functionality in the receiver (client).

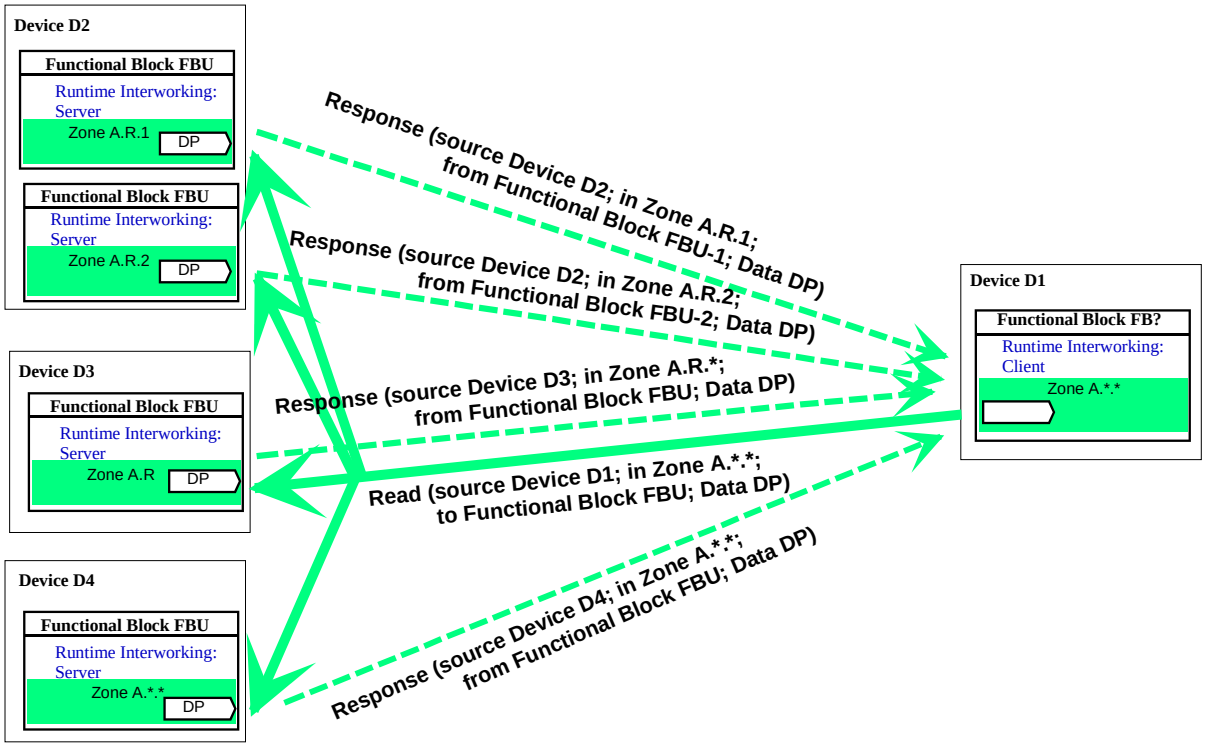


Figure 11 - Example: Read/Response using Wildcard zoning

“Sniffer” functionality in the Receiver

As described in clause 5.1.5, messages containing explicit structured (nested) zoning information are usually received (at least at the level of zoning, but not necessarily at Datapoint level) by Functional Blocks / devices belonging to the higher level zone too. Sniffer functionality is application specific and must be defined per Datapoint.

- Sniffer functionality is applicable to Datapoint information transmitted using InfoReport or Response service
- Sniffer functionality is not applicable to Datapoint information transmitted using Write or Read service ⇒ Wildcard addressing is more appropriate

Example: Sniffer and Wildcard mechanisms with structured zone of type X.Y.Z:

Each Functional Block will generate a “receiver zone address table” depending on its own configured zone and its Datapoints and related services. The following examples show the usual mechanisms. Details must be specified in the Application Specifications

Example: InfoReport or Response messages could be accepted by the receiver in the following zones

	Zoning information in received message							
Configured Zone in the receiver FB	X . Y . Z	X . Y . *	X . * . Z	X . * . *	* . Y . Z	* . Y . *	* . *. Z	*. * .*
X (. * . *)	√ Sniffer on Y.Z	√ Sniffer on Y	(√) Wildcard on Y Sniffer on Z	√ own zone	---	---	---	√ Wildcard for all zone
X . Y (. *)	√ Sniffer on Z	√ own zone	(√) Wildcard on Y Sniffer on Z	√ Wildcard on Y	---	(√) Wildcard on X	---	√ Wildcard for all zone
X . Y . Z	√ own zone	√ Wildcard on Z	(√) Wildcard on Y	√ Wildcard on Y.Z	(√) Wildcard on X	(√) Wildcard on X and Z	(√) Wildcard on X and Y	√ Wildcard for all zone

(√) : very limited usage in real applications

Example: Write or Read messages could be accepted by the receiver in the following zones

	Zoning information in received message							
Configured Zone in the receiver FB	X . Y . Z	X . Y . *	X . * . Z	X . * . *	* . Y . Z	* . Y . *	* . *. Z	*. * .*
X (. * . *)	---	---	---	√ own zone	---	---	---	√ Wildcard for all zone
X . Y (. *)	---	√ own zone	---	√ Wildcard on Y	---	√ Wildcard on X	---	√ Wildcard for all zone
X . Y . Z	√ own zone	√ Wildcard on Z	√ Wildcard on Y	√ Wildcard on Y.Z	√ Wildcard on X	√ Wildcard on X and Z	√ Wildcard on X and Y	√ Wildcard for all zone

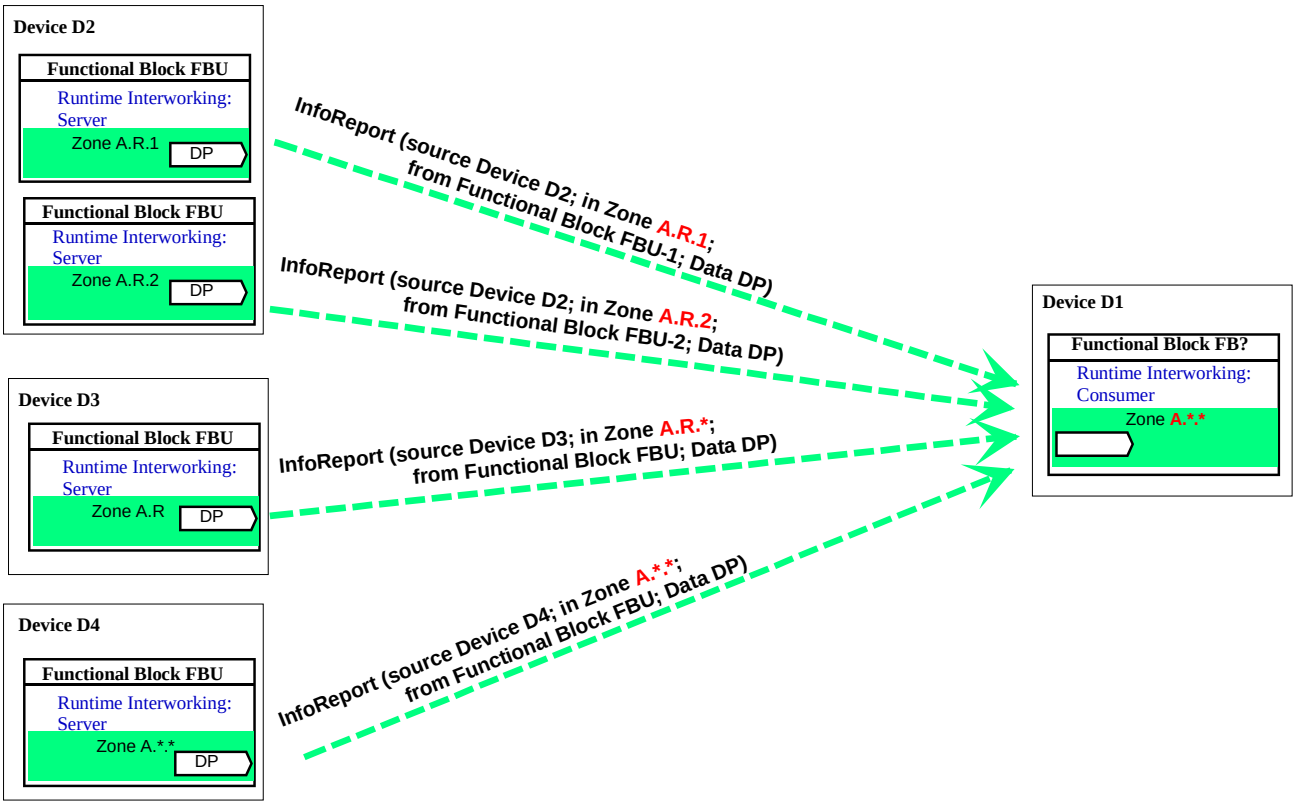


Figure 12- Example: “Sniffer” mechanism on InfoReport in structured zones

In the example above the Receiver of the Datapoint DP is a “Sniffer” within apartment A.

6.3.6 Handling of zone addresses and Datapoint access

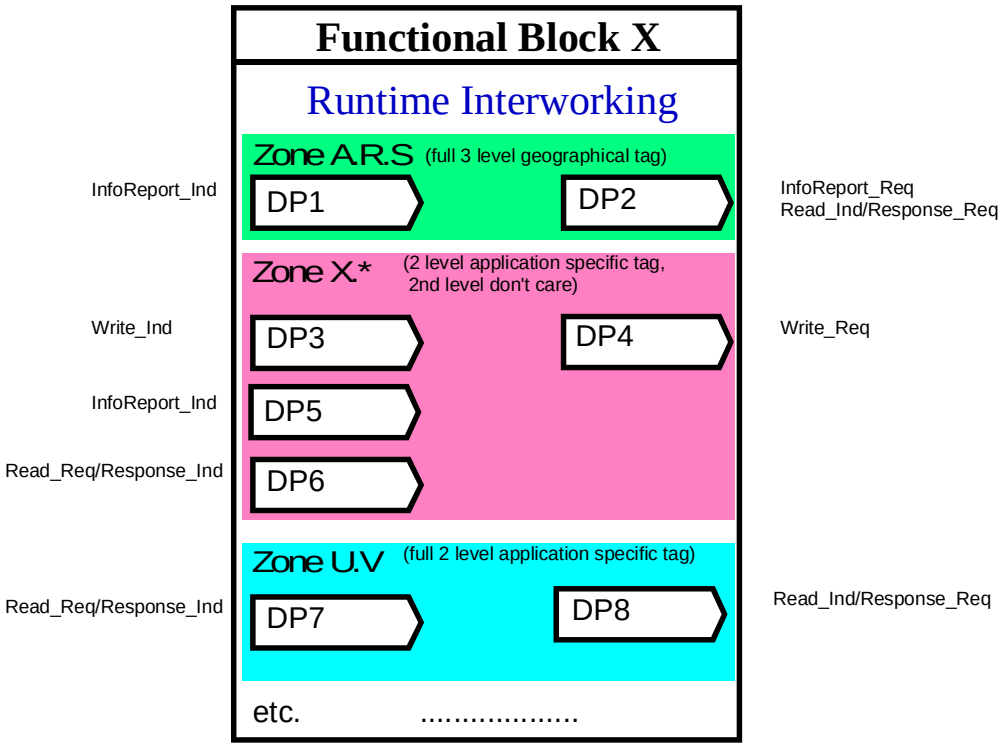


Figure 13- Example: Functional Block with data-interface in 3 zones

The Functional Block in the figure above has a data-interface with Datapoints in 3 different structured zones. For each Datapoint the supported LTE AL services are indicated.

- the output DP on the right are distributed spontaneously by InfoReport (request) or Write (request) service or may be read by Read (indication) service which results in a Response (request)
- the input Datapoints on the left are either spontaneously received messages containing the service InfoReport (indication) or Write (indication) or are polling inputs.
  If the Functional Block reads a Datapoint (Datapoint client, e.g. DP7) a Read (request) is sent spontaneously by the Functional Block which may result in one or more received Response (indication).

Example:

The table below lists for each Datapoint the supported LTE AL service primitives and the corresponding supported zones from the point of view of Functional Block X in Figure 13.

Datapoint	**Datapoint access**	**DP Ident**	**supported Zone(s)**		
	AL service primitive	**Functional Block / Datapoint**	**Sender**	**Receiver**	
Input DP1	InfoReport_Ind	FB_External_A / DP_External_a		**A.R.S** A.R.* A.*.S A.*.* *.R.S *.R.* *.*.S *.*.*	**geographical Zone List [1]** own zone and all wildcard combinations
Output DP2	InfoReport_Req	FBX / DP2	**A.R.S**		
	Read_Ind	FBX / DP2		**A.R.S** A.R.* A.*.S A.*.* *.R.S *.R.* *.*.S *.*.*	**geographical Zone List [1]** own zone and all wildcard combinations
	Response_Req	FBX / DP2	**A.R.S**		
Input DP3	Write_Ind	FBX / DP3		**X.*** *.*	**appl. spec. Zone List [1]** own zone and wildcard
Output DP4	Write_Req	FB_External_B / DP_External_b	**X.***		
Input DP5	InfoReport_Ind	FB_External_C / DP_External_c		**X.*** X.Y Sniffer *.*	**appl. spec. Zone List [2]** own zone and Sniffer on Y and wildcard
Input DP6 (Polling)	Read_Req	FB_External_D / DP_External_d	**X.***		
	Response_ind	FB_External_D / DP_External_d		**X.*** X.Y Sniffer (*.*) [1])	**appl. spec. Zone List [2]** own zone and Sniffer on Y (and wildcard)
Input DP7 (Polling)	Read_Req	FB_External_E / DP_External_e	**U.V**		
	Response_ind	FB_External_E / DP_External_e		**U.V** (U.*) [1]) (*.V) [1]) (*.*) [1])	**appl. spec. Zone List [3]** own zone (and all wildcard combinations)
Output DP8 (Polling)	Read_Ind	FBX / DP8		**U.V** U.* *.V *.*	**appl. spec. Zone List [3]** own zone and all wildcard combinations
	Response_Req	FBX / DP8	**U.V**		

[1]) responses with this zoning information would be the result of a incorrect behaviour of the server - but would be accepted by the client

The following paragraphs give an informal description of LTE-HEE message handling (the LTE-HEE AL service specification will contain a more formal description)

Concerning **zone address handling** the example shows that:

- For the **sending part** of a device the application program knows and determines for each Datapoint the zone in which the message is sent.
- Zone handling in the **receiving part** of a device is more complex because of Wildcard and Sniffer capabilities. Message acceptance criteria depend on the Datapoint and the access to it (AL service primitive) and on the zoning information. For each Datapoint and access mechanism a specific “Zone List” in the receiver is associated. Zone acceptance is checked in the Application Interface Layer.
- For each Functional Block an LTE Group Address table is generated out of this set of “Zone List”. The LTE Group Address table is used in the receiver (Link Layer) to accept or reject received LTE group messages.
- If an LTE group message is accepted for its zoning information (at link layer level) additional checks are necessary to detect if access to a given Datapoint (Functional Block, DP, AL service primitive) is allowed in this zone. This checking procedure depends on other Datapoint access tables and is executed in the Application Interface Layer.

Concerning **Datapoint access handling** the example shows that:

- For the **‘server’** part (Read.ind / Response.req, Write.ind) the Functional Block needs a list of its own Datapoints which can be accessed by a client. For an incoming Read.ind or Write.ind, besides zoning information also the Datapoint address and access rights must be checked (in the Application Interface Layer) and in case of Read service the corresponding Response must be generated by the application program.
- For the **‘client’** part (Read.req / Response.ind, Write.req) the Functional Block has a list of external Datapoints which shall be accessed by Read/Write service. For each Read.req or Write.req besides zoning information also the external Datapoint address must be known in the client Functional Block. For each received Response.ind, besides zoning information also the external Datapoint address must be checked (in the Application Interface Layer)
- For **input** Datapoints accessed by InfoReport.ind service the Functional Block needs a list of external Datapoints which shall be consumed by the Functional Block. For each InfoReport.ind, besides zoning information also the external Datapoint address must be known and checked.
- Concerning **output** Datapoints distributed by InfoReport.req service the Functional Block has a list of its own Datapoints which shall be distributed by this mechanism.

6.4 LTE device model

A Device consists of one or more Functional Blocks which are represented in the LTE-HEE protocol by **Interface Objects.** Datapoints of a Functional Block are represented as **Properties** of the Interface Object.

LTE-HEE Interworking and the implementation of the corresponding Functional Blocks, properties and zoning information is part of the KNX Certification of a product.

Note: If Datapoints are in a 1:1 relationship and if the sending and receiving functional blocks are located in the same device, it's not meaningful that the messages from the producer to the consumer appeare on the bus.

LTE devices provide an additional defined and standardised **S-Mode Interface** (double implementation of Datapoints) which enables Interworking with other Modes (e.g. S-Mode, Controller-Mode).

The implementation of the S-Mode Interface is relevant for Certification

The S-Mode Interface shall contain at least the mandatory Datapoints of a Functional Block. A broader S-mode interface may be reasonable for specific applications. I.e. if an optional Datapoint is implemented it shall also be available in S-Mode. This is defined by the Application Specification. A manufacturer is free to implement other optional Datapoints in S-Mode which will be tested for certification.

The Datapoint format / type in LTE-HEE messages and standard messages may be different

- In the standard messages, existing Datapoint Types should be used to enable general purpose Interworking
- In an LTE-HEE environment, specific application needs may lead to "richer" and more complex data structures which allow higher functionality but may not be used in the standard system.
- If for a given Datapoint LTE-HEE and standard system use a different Datapoint format, data must be converted at device level. Conversion of a structured Datapoint from "richer" LTE-HEE format to standard format may lead to multiple Datapoints in S-mode or loss of information and / or inconsistency of data.

The "double implementation" of Datapoints in LTE-HEE and S-Mode leads to the situation that:

- An **output Datapoint** is sent twice if a COV or heartbeat event occurs. Decomposition of a structured LTE-HEE Datapoint into more than one S-Mode Datapoint will trigger multiple S-Mode messages. $\Rightarrow$ this creates more traffic on the bus
- For **input Datapoints** with 1:1 or 1:N relationship (only one source of the input) either the LTE-HEE input or the S-Mode input will be active and the S-Mode input will usually have priority
- For **input Datapoints** with N:1 relationship (multiple source for one input) the LTE-HEE input as well as the S-Mode input will be active
- The detailed handling of "double implementation" of Datapoints is application specific and shall be specified in the Application Interworking Standards.

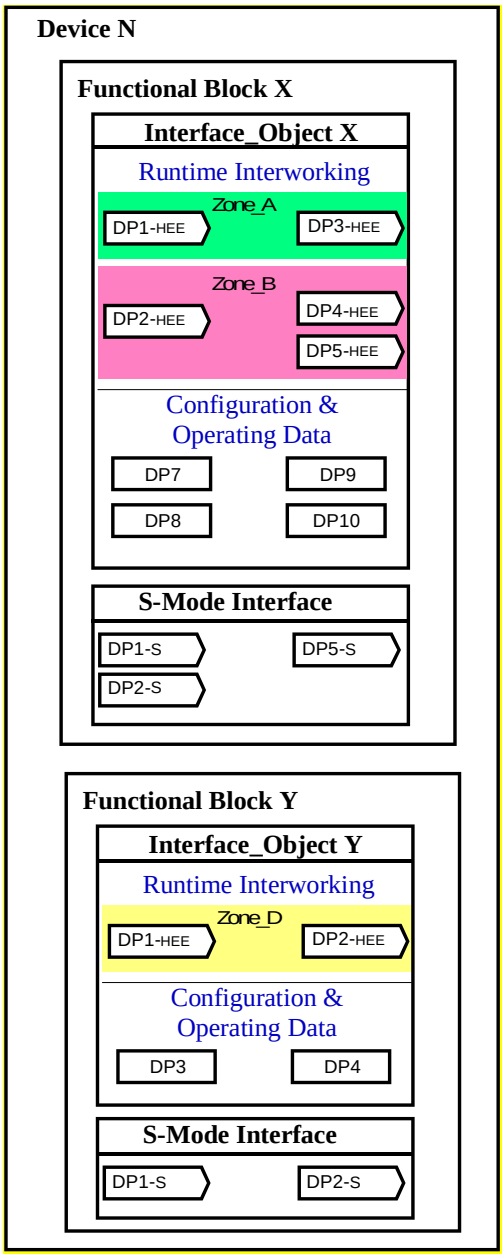


Figure 14 - Example: Device with 2 Functional Blocks containing LTE-HEE and S-Mode interface

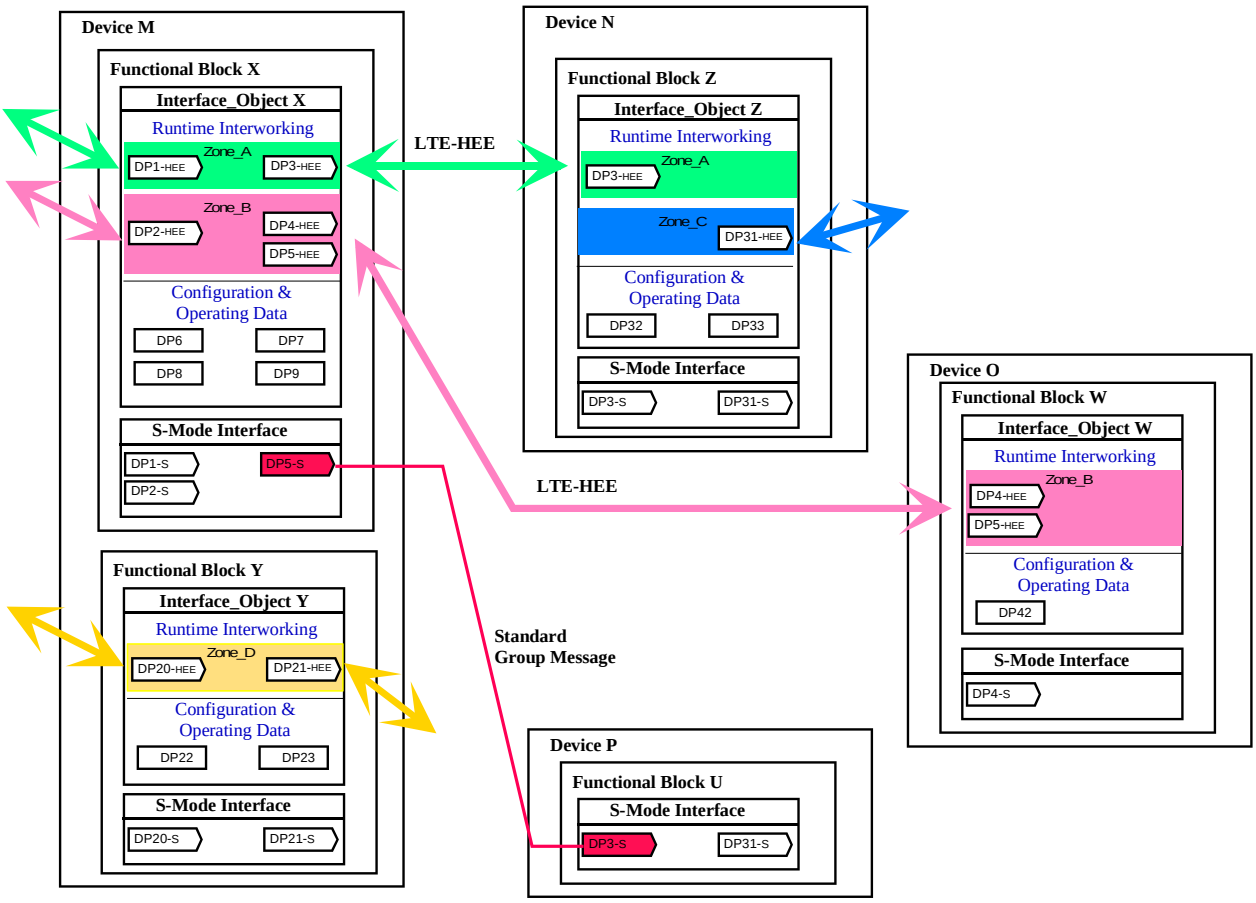


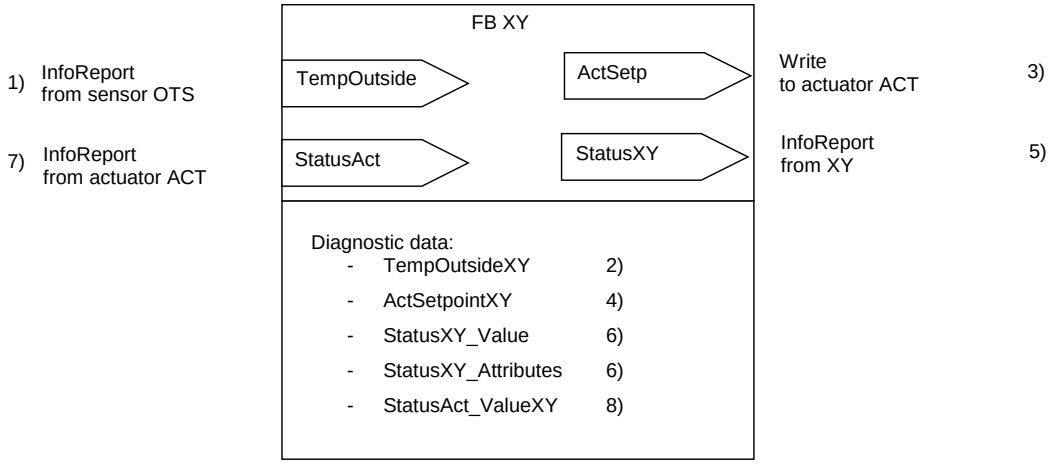
Figure 15 - Example: Interworking within LTE-HEE and S-Mode

6.5 Properties: LTE-HEE Runtime data versus Diagnostic data

The following example shows the difference between LTE-HEE properties used for runtime-Interworking and Properties like diagnostic data.

As explained before, for LTE-HEE runtime Interworking only local data (i.e. own data) of a Functional Block is addressed by the corresponding Interface Object whereas external data is addressed by the remote Interface Object.

This has some consequences concerning the mapping of runtime Interworking data to diagnostic data.



1) This InfoReport input property from the outside temperature sensor OTS is not part of the Interface Object XY. It is an external property of OTS and can not directly be accessed as a property of XY. See also clause 6.3.1
   Data of this input could be made visible as a local diagnostic value TempOutsideXY of Interface Object XY ⇒ 2)

   IMPORTANT: Actions like e.g. 'Override' can not be executed directly on InfoReport inputs
   ⇒ use local representation 2) of the input instead

2) Local copy of TempOutside used within XY. This Property is part of Interface Object XY and can be accessed using Property services / individual addressing.
   Example: TempOutsideXY can e.g. locally be 'overridden' using A_PropertyValue_Write service

3) This output controls the setpoint of the remote actuator ACT using LTE Write service. XY is the client of the Property ActSetp in the remote Interface Object ACT. This output can not be accessed directly as a Property of Interface Object XY. See also clause 6.3.2.
   If needed, data of this output can be made visible as a local diagnostic value ActSetpointXY of Interface Object XY ⇒ 4)

4) This Property of Interface Object XY is a diagnostic value which represents the locally calculated actual value of the actuator setpoint. In this example the value of Datapoint ActSetpointXY is in principle the source of the output value of ActSetp 3)
   ActSetp output is sent if ActSetpointXY has changed significantly (COV) or due to heartbeat repetition.

5) This InfoReport output Property is part of the Interface Object XY and it is sent spontaneously to "any" remote Interface Object. This Property can be accessed using LTE Read service or A_PropertyValue_Read with individual addressing.

6) LTE runtime Datapoints have often a structured data type. Parts of structured data used for runtime-Interworking like output 5) could case by case also be interesting as diagnostic value for visualisation. For visualisation, access to simple data types is generally more convenient. Therefore structured Datapoints may be split-up into multiple diagnostic data for visualisation
⇒ two Properties StatusXY_Value and StatusXY_Attributes

7-8) same as for 1) and 6) Only part of the structured InfoReport input data is accessible as local Property for visualisation.

In this example the Interface Object XY is the owner of the following Properties

Property Identifier	**Datapoint Name**	**Datapoint Type Name**	**Datapoint Type Code**
LTE-HEE process data (runtime Interworking, LTE group- and individual addressing)			
51	StatusXY	DPT_....	xxxx.yyyy
Parameters and Diagnostic Data (individual addressing only)			
...			
110	TempOutsideXY	DPT_....	....
111	ActSetpointXY	DPT_....	....
112	StatusXY_Value	DPT_....	....
113	StatusXY_Attributes	DPT_....	....
114	StatusACT_ValueXY	DPT_....	....
...			

Interface Object XY is the consumer/client of the following Datapoints:

Interworking Obj Type	**Property Identifier**	**Datapoint Name**	**Datapoint Type Name**	**Datapoint Type Code**
	LTE-HEE process data (runtime Interworking, LTE group addressing)			
OTS	51	TempOutside	DPT_....	.....
ACT	51	StatusAct	DPT_....	.....
ACT	52	ActSetp	DPT_....	.....

6.6 LTE Datapoints description

In the Functional Block specifications each Datapoint shall be described in a standardised way. For LTE-HEE – style Datapoints an extended Datapoint description is introduced which covers the specific features of LTE-HEE mechanisms. Four templates are defined: two for LTE-HEE inputs (client and server side) and two for LTE-HEE outputs (client and server side).

For the Datapoint description of the S-Mode interface, the standard Datapoint description template is used (as described in the general KNX Interworking Model)

6.6.1 LTE-HEE Client Input (InfoReport, Read-Response)

Template (including real HVAC-HWH example ValueDemBOC Datapoint): see also Figure 4 and Figure 6

FB: BUC	LTE Client Input Name: ValueDemBOC	Mandatory ☒ Optional ☐
Description: This input signal contains the actual burner control information from the BoC. It is used to control the stages/modulation grade of one burner		

DPT:	Name	DPT_ValueDemBOC	DPT ID	207.102	Datatype format	U_8B_8

Field	Description	Sup.	Unit	Default
RelBurnerDem	Relative demand %: for modulating burner	O	%	cs
Attributes	Bitset containing control info			
– Stage1Control	controls operation of stage 1 or base stage	M	bool on/off	cs
– Stage2Control	controls stage 2 for two stage burner	O	bool on/off	cs

Communication:			
Binding Group:			
Class	Type	Default	
Geographical ☐			
Application Specific ☒	ProdSegmH.Producer	1.1	
Unassigned ☐	Broadcast ☐ Configurable ☐		
DP Address:	IO Type(ID): 129 (BOC)	Property ID: 53	
LTE-Service (event):	InfoReport Sniffer on Binding Group: --		
InfoReport ☒	Timeout: 3	Min	
LTE-Service (polling): Read – Response ☐	Read Wildcard / Resp Sniffer on Binding Group: --		
Value after Powerup: Default Value ☒ Stored Value ☐			
Exception Handling:		Save at Powerdown ☐	
The burner controller will use a company specific default value after power-up or in case of communication failure, if no data from BoC is received (normally burner off). Actual burner stage / modulation grade is also depending on safety related mechanisms within the burner controller			

Special Features:
....

- **FB:**

  This field shall contain the Functional Block for which the Datapoint is an input.
  e.g. BUC = Burner Controller

- **LTE Client Input Name:**

  This field shall contain the name of the Datapoint

- **Mandatory/Optional:** Check either

  Mandatory: this Datapoint shall be implemented or

  Optional: this Datapoint may be implemented; if implemented it shall be implemented in the specified way

  Add a footnote in the field “Special Features” if implementation of this input may depend on special conditions

- **Description:**

  This field shall contain the purpose of this input Datapoint and summarises the usage of the Datapoint in the Functional Block.

- **DPT:**

  Name: These are the Name and ID of the Datapoint Type that shall be used for this Datapoint. The name starts with "DPT_". e.g. DPT_ValueDemBOC

  ID: The DPT ID is composed of 2 16-bit fields separated by a dot. Both are filled in by the Designer and shall refer to an accepted Datapoint. If such a Datapoint Type does not yet exist, this field shall be left open and the Functional Block proposal shall be accompanied by an application for a new Datapoint Type.

  Datatype format: Indicates the octet format and order of fields for this DPT; specially for compound DPT
  e.g. U8B8 : compound Datapoint: 1st field unsigned integer 8bit, 2nd field: bit set 8bit

  Field: Datapoint-field in compound Datapoint Types: each field shall be described individually

  Description: Short description of Datapoint-field: type of data and usage

  Sup.: In compound Datapoint Types the support of individual fields is usually mandatory (M) but may be optional (O) or conditional (C) in some cases.
  Optional support is e.g. often the case for attributes.
  E.g. a one stage burner will ignore the control information for a 2 stage or modulating burner.

  Unit: This is the unit of the individual field within the Datapoint Type. It is only repeated here for better understanding of the Datapoint description.

  Default: Default value of each individual field of the Datapoint. The default value may be used for initialisation after power up or in case of receiver timeout. The default value may be a constant value or may depend on a parameter. The value “cs” means “company specific”

- **Communication – Binding Group:**

  Class: type of LTE zoning information. One box shall at least be checked. For e.g. multipurpose sensors / actuators more than one type of zones may be possible
  (e.g. check Unassigned and Application Specific -> DistrSegmH for a Flow Temperature Sensor)

  Type: indicate detailed type of LTE zoning information for Geographical and Application Specific zones.
  e.g. A.R.S or A.R.* for Geographical zone or ProdSegmH.Producer for Application Specific zone

  For Unassigned zone check “broadcast” (fixed) or “configurable”

  If the receiver can be a Sniffer on structured zones this is indicated with **(n)** after the name of the Binding Group; e.g. ProdSegmH.Producer (n)

  Default: default zoning number at manufacture to enable Plug & Play operation in simple applications. The default value may be fixed or company specific “cs”

- **Communication – DP Address:**

  The combination of the Interface Object Type and Property ID allow for unambiguous interpretation of the property.

  IO Type (ID): Interface Object Type of the remote Functional Block (Datapoint source)

  ⇒ Numeric value (e.g. 129) and name (e.g. BOC)

  Property ID: Property identifier (numeric value) of the Property

- **Communication – LTE-Service (event):**

  InfoReport: check if spontaneous reception is supported.

  InfoReport Sniffer on Binding Group: In case of InfoReport input the receiver may support Sniffer functionality on zoning information. Whether Sniffer functionality is allowed or not depends on the application model. In case of Sniffer capability indicate the corresponding zone; e.g. ProdSegmH.Producer (n)

  Remark: Sniffer vs. Wildcard
  InfoReport input messages may always contain Wildcard zoning information and the receiver must react. This is a generic mechanism and part of the LTE model and no indications in the table are necessary.

  Timeout: Receiver timeout: fill in only for InfoReport input signals which are provided also by heartbeat-repetition.
  Rule: timeout = 2 x heartbeat period of the sender + Δt; Δt = 1 minute (fixed for LTE)

- **Communication – LTE-Service (polling):**

  InfoReport: Check only if input is a polling input value; i.e. the functional block is a Datapoint client which polls this Datapoint from a remote functional block. I.e. the client sends a Read (request) and receives none; one or multiple Response messages (indications)

  Read Wildcard / Resp Sniffer on Binding Group: The client may support Wildcard addressing on zoning information in the Read (request) in order to poll multiple zones efficiently with one message (beware of Bus-overload !). The result may be multiple Response messages (indications) from different zones and the receiver shall in this case behave as a Sniffer on these zones to get all Response messages. Whether Wildcard / Sniffer functionality is allowed or not for a given polling input depends on the application model. In case of Wildcard / Sniffer capability indicate the corresponding zone; e.g. ProdSegmH.**(n)**

- **Value after power-up**

  This field shall contain the value of the input after power-up:
  check either ‘default value’ (see DPT) or ‘stored value’ if the Datapoint value is saved at power-down.

- **Exception Handling:**

  This filed contains the description of special mechanisms like behaviour in case of no / disturbed communication or power down etc.

  Check ‘Save at Powerdown’: if last Datapoint input value shall be saved at power down

- **Special Features:**

  This filed contains e.g. “footnotes” in order to give more detailed explanation of features if necessary.

6.6.2 LTE-HEE Server Input (Write)

Template (including real HVAC-HWH example FuelSelect Datapoint): see also Figure 5

FB: BUC	LTE Server Input Name: FuelSelect	Mandatory ☐ Optional ☒
Description: This input allows switching between different fuel options and contains the type of fuel to be used by the BUC. This information may be written by the BoilerController.		

DPT:	Name	DPT_FuelType	DPT ID	20.100	Datatype format	N_8

Field	Description	Sup.	Unit	Default
FuelSelect	see above	M	enum	cs

Communication:		
Binding Group:		
Class	Type	Default
Geographical ☐		
Application Specific ☒	ProdSegmH.Producer	1.1
Unassigned ☐	Broadcast ☐ Configurable ☐	
DP Address:	IO Type(ID): 128 (BUC)	Property ID: 53
LTE-Service (event): Write ☒	Timeout: --[2)]	Min
Property-Service (individual access):	Read only ☐ Read/Write ☒	
Value after Powerup: [1)] Default Value ☒ Stored Value ☒		
Exception Handling:		Save at Powerdown[1)] ☒

[1)] The burner controller will use a company specific default value or stored value after power-up (company specific behaviour)

[2)] This signal has no heartbeat ⇒ last value from BOC is kept until next update or default after power-up

Special Features:
--

- **FB:**

  See clause 6.6.1

- **LTE Server Input Name:**

  See clause 6.6.1

- **Mandatory/Optional:**

  See clause 6.6.1

- **Description:**

  See clause 6.6.1

- **DPT:**

See clause 6.6.1

- **Communication – Binding Group:**

  Class: see clause 6.6.1

  Type: indicate detailed type of LTE zoning information for Geographical and Application Specific zones.
  e.g. A.R.S or A.R.* for Geographical zone or ProdSegmH.Producer for Application Specific zone

  For Unassigned zone check “broadcast” (fixed) or “configurable”

  Remark: Write inputs do NOT support Sniffer mechanism

  Default: default zoning number at manufacture to enable Plug & Play operation in simple applications. The default value may be fixed or company specific “cs”

- **Communication – DP Address:**

  The combination of the Interface Object Type and Property ID allow for unambiguous interpretation of the property.

  IO Type (ID): Interface Object Type of the Functional Block (local Datapoint server)

  ⇒ Numeric value (e.g. 128) and name (e.g. BUC)

  Property ID: Property identifier (numeric value) of the Property

- **Communication – LTE-Service (event):**

  Write: check if spontaneous reception is supported (this box shall always be checked because there is no other mechanism described by this table)

  Timeout: Receiver timeout: fill in only for Write input signals which are provided also by heartbeat-repetition.
  Rule: timeout = 2 x heartbeat period of the sender + Δt; Δt = 1 minute (fixed for LTE)

- **Communication – Property-Service (individual access):**

  Since this input is a Property of the Functional Block it can be accessed also by standard Property Services using individual addressing

  Read only: check if only A_PropertyValue_Read service is supported

  Read/Write: check if A_PropertyValue_Read and A_PropertyValue_Write services are supported

- **Value after power-up**

  See clause 6.6.1

- **Exception Handling:**

  See clause 6.6.1

- **Special Features:**

  See clause 6.6.1

6.6.3 LTE-HEE Server Output (InfoReport, Read-Response)

Template (including real HVAC-HWH example StatusBUC Datapoint): see also Figure 4 and Figure 6

FB: BUC	LTE Server Output Name: StatusBUC	Mandatory ☒ Optional ☐
Description: This output process signal contains status information of the burner to be used in the BoC for boiler control		

DPT:	Name	DPT_StatusBUC	DPT ID	207.100	Datatype format	$U_8U_{16}U_{16}B_8$

Field	Description	Sup.	Range	Unit	COV	Default
PrelBurner [1])	Actual relative power	O	0..100%	%	10%	0%
OpHrsBurnerStage1	Operating hours (stage1/base st)	M	0..65535	h	--	--
OpHrsBurnerStage2	Operating hours (stage 2)	O	0..65535	h	--	--
Attributes	Bitset containing status info					
– Fault	burner failure	M	fault/ok	bool	Y	ok
– Stage1	stage 1 or base stage active	M	true/false	bool	Y	false
– Stage2	stage 2 active	O	true/false	bool	Y	false
– PrelBurnerValid	validity of PrelBurner Field	M	true/false	bool	Y	false
– OpHrsBurnerSt1Valid	validity of OpHrsBurnerStage1 Field	M	true/false	bool	Y	false
– OpHrsBurnerSt2Valid	validity of OpHrsBurnerStage2 Field	M	true/false	bool	Y	false

Communication:			
Binding Group:			
Class	Type		Default
Geographical ☐			
Application Specific ☒	ProdSegmH.Producer		1.1
Unassigned ☐	Broadcast ☐ Configurable ☐		
DP Address:	IO Type(ID): 128 (BUC)	Property ID: 51	
LTE-Services (event):	COV ☒ MinRepTime: 10 sec	Heartbeat: 1 min	
InfoReport ☒ (LTE Read-Response polling of the output shall always be supported)	Output per default communicating ☐	Binding Group Wildcard allowed ☐	
	Tx Prio: High ☐ Normal ☒ Low ☐		
	Transm after Powerup: Stored Value ☐ Act Value ☒ Default Value ☐		
Property-Service (individual access):	Read only ☒ Read/Write ☐		
Exception Handling:		Save at Powerdown ☐	
Special Features:			

[1]) value for 1 stage burner: void ⇒ PrelBurnerValid Flag=false
value for 2 stage burner: some 2 stage BuC may be able to indicate % value if only stage 1 is on. But this is an optional feature: This field is optional for 2 stage burner ⇒ validity according to PrelBurnerValid flag
COV / transmission condition: local time of the device matches a schedule's switching point.

- **FB:**

  This field shall contain the Functional Block for which the Datapoint is an output.
  e.g. BUC = Burner Controller

- **LTE Server Output Name:**

  This field shall contain the name of the Datapoint

- **Mandatory/Optional:** Check either

  Mandatory: this Datapoint shall be implemented or

  Optional: this Datapoint may be implemented; if implemented it shall be implemented in the specified way

  Add a footnote in the field "Special Features" if implementation of this output may depend on special conditions

- **Description**

  This field shall describe the functionality and purpose of this output Datapoint

- **DPT:**

  Name: see clause 6.6.1

  ID: see clause 6.6.1

  Datatype format: see clause 6.6.1

  Field see clause 6.6.1

  Description: see clause 6.6.1

  Sup.: In compound Datapoint types the support of individual fields is usually mandatory (M) but may be optional (O) or conditional (C) in some cases. Optional support is often the case for attributes.
  E.g. a one stage burner will not provide status information of a 2 stage or modulating burner.
  If not supported the fields shall contain the Default value !

  Range: Recommended output range of the individual field.
  Remark: the receiver must always be able to handle the full range of the Datapoint Type.

  Unit: This is the unit of the individual field within the Datapoint Type. It is only repeated here for better understanding of the Datapoint description.

  COV: This is the COV condition (change of value) of the individual field which triggers a spontaneous transmission of the Datapoint. If the output is not sent spontaneously due to COV this information is void.

  Default: Default value of each individual field of the Datapoint.
  Used: - if default value shall be transmitted immediately after power up before up to date data is available from the application
  - for unsupported Datapoint-fields (void value)

- **Communication – Binding Group:**

  Class: see clause 6.6.1

  Type: indicate detailed type of LTE zoning information for Geographical and Application Specific zones. Zone Wildcard addressing is allowed.
  e.g. A.R.S or A.R.* for Geographical zone or ProdSegmH.Producer for Application Specific zone

  For Unassigned zone check “broadcast” (fixed) or “configurable”

  Default: default zoning number at manufacture to enable Plug & Play operation in simple applications. The default value may be fixed or company specific “cs”

- **Communication – DP Address:**

  The combination of the Interface Object Type and Property ID allow for unambiguous interpretation of the property.

  IO Type (ID): Interface Object Type of the sending Functional Block (Datapoint source)
  ⇒ Numeric value (e.g. 128) and name (e.g. BUC)

  Property ID: Property identifier (numeric value) of the Property

- **Communication – LTE-Service (event):**

  InfoReport: check if spontaneous transmission is supported.

  If this box is not checked, the output is a pure polling output (i.e. LTE Read access)
  Remark: LTE Read access to an InfoReport output is always possible. This is a feature of the LTE application model and therefore no further indications in the table are necessary.

  COV: Check if Datapoint is transmitted spontaneously due to change of value condition as described in DPT.COV

  MinRepTime: Minimum repetition time of the signal if multiple COV conditions occur within a short time. This time defines is the minimum update interval on the bus and the max. traffic a Datapoint may produce. Value depends on the needed “sampling rate” for a given application. The value is usually fixed in LTE systems but may also depend on a parameter.

  Heartbeat: Periodical repetition of the signal if no COV condition occurs. If the Datapoint has no heartbeat the field is void (--).
  For LTE systems the heartbeat period is fixed for each Datapoint (as well as the corresponding receiver timeout)

  Output per default communicating: If multiple Functional Blocks are combined in one device, communication of some Datapoints may be purely device-internal. However all respective mandatory Datapoints of the separate Functional Blocks shall be separately and fully implemented with corresponding independent behaviour. The following limitation is however allowed: The spontaneous communication may be disabled. This means that the Datapoint is present but not communicating. It shall be possible to re-enable the spontaneous communication.

  Check if the output Datapoint is per default communicating on the network.

  Binding Group Wildcard allowed: Check if spontaneous transmission using InfoReport service may contain Wildcard zoning information. Whether Wildcard addressing is allowed or not depends on the application model.

  Remark: In the LTE polling service Read – Response the Read (indication) may always contain Wildcard zoning information and the server must react. However after a Wildcard Read the corresponding Response message shall contain precise zoning information. This is a generic mechanism and part of the LTE model.

  Tx-Prio: Transmission priority for InfoReport service: usually normal priority

  Transm. after Powerup: Value to be transmitted after power up.

Check ‘Stored Value’ if the Datapoint shall be transmitted immediately after power up containing the stored value before power down

Check ‘Act Value’ if the application shall wait and calculate a new actual value after power up to be transmitted only if available

Check ‘Default Value’ if the Datapoint shall be transmitted immediately after power up containing the default value as described in DPT.Default

- **Communication – Property-Service (individual access):**

  Since this output is a Property of the Functional Block it can be accessed also by standard Property Services using individual addressing

  Read only: check if only A_PropertyValue_Read service is supported

  Read/Write: check if A_PropertyValue_Read and A_PropertyValue_Write services are supported

- **Exception Handling:**

  This field contains the description of special mechanisms like behaviour in case of errors etc. (e.g. behaviour of a sensor output in case of sensor failure)

  Check ‘**Save at Powerdown’**: if Datapoint value shall be saved at power down

- **Special Features:**

  This filed contains e.g. “footnotes” in order to give more detailed explanation of features if necessary.

6.6.4 LTE-HEE Client Output (Write)

Template (including real HVAC-HWH example FuelSelect Datapoint): see also Figure 5

FB: BOC	LTE Client Output Name: FuelSelect	Mandatory ☐ Optional ☒

Description:

This Datapoint is used by the BOC to switch between different fuel options in the BUC. The BOC knows the supported fuel types according to the BurnerSpec information

DPT:	Name	DPT_FuelType	DPT ID	20.100	Datatype format	N_8

Field	Description	Sup.	Range	Unit	COV	Default
FuelSelect	see above	M	[1-3]	enum	Y	cs

Communication:

Binding Group:

Class	Type	Default
Geographical ☐		
Application Specific ☒	ProdSegmH.Producer	1.1
Unassigned ☐	Broadcast ☐ Configurable ☐	
DP Address:	IO Type(ID): 128 (BUC) Property ID: 53	
LTE-Services (event):	COV ☒ MinRepTime: -- sec Heartbeat: -- min	
Write ☒	Output per default communicating ☐ / Binding Group Wildcard allowed ☐	
	Tx Prio: High ☐ Normal ☒ Low ☐	
	Transm after Powerup: Stored Value ☐ Act Value ☒ Default Value ☐	

Exception Handling:	Save at Powerdown ☐
--	

Special Features:

--

- **FB:**

  This field shall contain the Functional Block for which the Datapoint is an output.

- **LTE Client Output Name:**

  This field shall contain the name of the Datapoint

- **Mandatory/Optional:**

  See clause 6.6.3

- **Description**

  See clause 6.6.3

- **DPT:**

  See clause 6.6.3

- **Communication – Binding Group:**

  See clause 6.6.3

- **Communication – DP Address:**

  The combination of the Interface Object Type and Property ID allow for unambiguous interpretation of the property.

  IO Type (ID): Interface Object Type of the remote Functional Block (Datapoint server)
  ⇒ Numeric value (e.g. 128) and name (e.g. BUC)

  Property ID: Property identifier (numeric value) of the Property

- **Communication – LTE-Service (event):**

  Write: check if spontaneous transmission using LTE Write service is supported (this box shall always be checked because there is no other mechanism described by this table).

  COV: Check if Datapoint is transmitted spontaneously due to change of value condition as described in DPT.COV

  MinRepTime: Minimum repetition time of the signal if multiple COV conditions occur within a short time. This time defines is the minimum update interval on the bus and the max. traffic a Datapoint may produce. Value depends on the needed "sampling rate" for a given application.
  The value is usually fixed in LTE systems but may also depend on a parameter. If the field is void (--) e change of value is written to the network.

  Heartbeat: Periodical repetition of the signal if no COV condition occurs. If the Datapoint has no heartbeat the field is void (--).
  For LTE systems the heartbeat period is fixed for each Datapoint (as well as the corresponding receiver timeout)

  Output per default communicating: see clause 6.6.3

  Binding Group Wildcard allowed: Check if spontaneous transmission using Write service may contain Wildcard zoning information. Whether Wildcard addressing is allowed or not depends on the application model.

  Tx-Prio: Transmission priority for Write service: usually normal priority

  Transm. after Powerup: see clause 6.6.3

- **Exception Handling:**

  This field contains the description of special mechanisms like behaviour in case of errors etc.

  Check '**Save at Powerdown'**: if Datapoint value shall be saved at power down

- **Special Features:**

  This filed contains e.g. "footnotes" in order to give more detailed explanation of features if necessary.

6.6.5 Property ID range

For LTE-HEE property addressing a preferred range of Property ID is foreseen.

The following table is a guideline for LTE-HEE property addressing

Property ID range	Usage
00.50	RESERVED for standard properties
51..100	standard LTE-HEE properties for runtime Interworking numbering: upwards starting with 51
101..109	LTE zone parameters (recommended range) value to be accessed using individual addressing numbering: upwards starting with 101
110..154	standard parameters and diagnostic value to be accessed using individual addressing numbering: upwards starting with 110
155..254	company specific (private) parameters and diagnostic value to be accessed using individual addressing numbering: downwards starting with 254
255	Escape for LTE-HEE private properties for runtime Interworking

6.7 LTE runtime communication on KNX-RF

6.7.1 General

LTE runtime communication on KNX-RF shall use the same basic LTE-mechanisms and services as on TP or PL media, this is, zoning information shall be mapped to LTE Group Addresses and Datapoint addressing shall be based on Interface Objects and Property mechanisms.

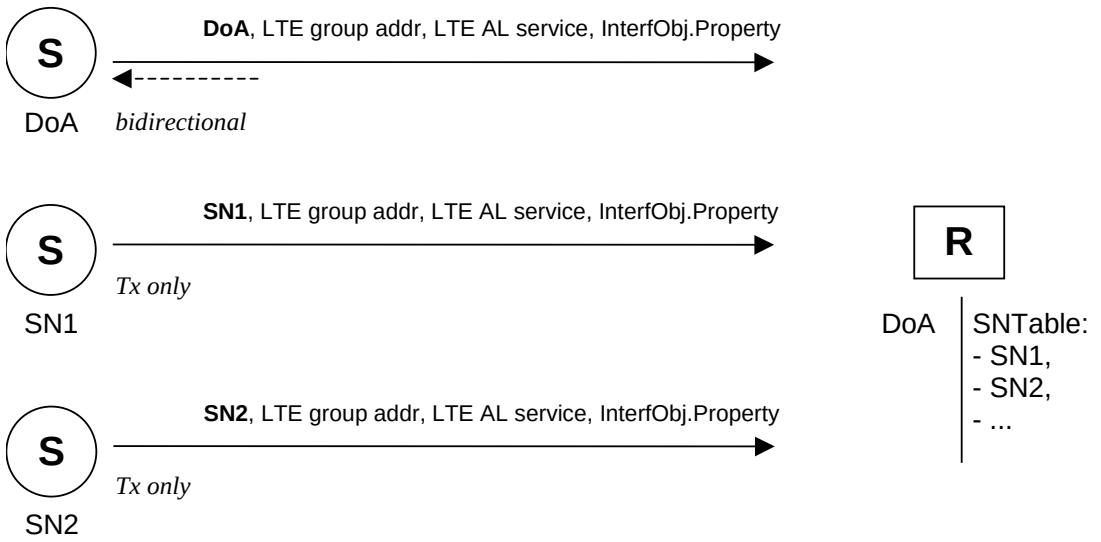
On KNX-RF LTE Group Addresses shall in addition be extended by a unique identifier to guarantee separation of neighbouring systems. The extended LTE Group Address shall contains either the KNX Serial Number of the sender (transmit-only devices) or a unique Domain Address (bidirectional devices).

The sender shall include either its own KNX Serial Number or the Domain Address in the LTE frame.

In the receiver the KNX Serial Number or the Domain Address shall be evaluated according to the AddrExtensionType bit in the L/NPCI.

The Data Link Layer in the receiver shall apply the following additional LTE frame acceptance criteria.

- In case the AddrExtensionType indicates that the SN/DoA-field contains a Domain Address, then the frame shall only be accepted if the Domain Address of the sender and receiver are the same, otherwise the frame shall be discarded.
- In case the AddrExtensionType indicates that the SN/DoA-field contains a KNX Serial Number, the frame shall only be accepted if the KNX Serial Number of the sender is contained in the KNX Serial Number Table of the receiver, otherwise the message shall be discarded.
- For further details see [02] clause “LTE Group Address Extension”.



S: Sender
R: Receiver
DoA: Domain Address
SN1: Serial Number of device 1
SN2: Serial Number of device 2
SNTable: Serial Number Table

Figure 16 - Extended LTE Group Addresses with Domain Address or KNX Serial Number

6.7.2 Bidirectional devices

Bidirectional LTE devices (LTE RF BD device) of the same KNX-RF installation shall share one common Domain Address. This Domain Address shall be the KNX Serial Number of one “master” device in the installation that shall distribute the Domain Address and Individual Address during "teach-in".

Please refer to 8.1 for the Management requirements of the Domain Address of LTE RF BD devices.

LTE messages of LTE RF BD devices shall contain the Individual Address of the sender.

Setting of the Individual Address: see please refer to clause 8.2.2.

As on TP or PL media, LTE messages from different devices within one Domain are differentiated by the LTE zoning information (LTE Group Addresses) and the Interface Object Type Property Identifiers. Therefore the proper LTE zoning information shall be configured on the sender and receiver in order to avoid zone conflicts.

Zoning information can be configured

- locally, depending on the local user interface of the device, or
- remotely via RF from a “master” device or a tool. In the LTE model, zoning information consists of Properties of the corresponding Functional Block/ Interface Object. In order to set zoning information via RF, Domain Address and Individual Address of the device must be assigned before.

Spontaneous LTE runtime communication using Domain Address shall be inactive

- ex-factory, void Domain Address 000000000000h,
- during “teach-in” configuration procedure.

LTE-Interworking with unidirectional (transmit-only) devices

Depending on the application program and device Profile, bidirectional devices shall also support LTE runtime Interworking with transmit-only devices. See clause 6.7.3.1.

6.7.3 Unidirectional devices

6.7.3.1 Transmit-only devices

Transmit-only devices cannot acquire the Domain Address of the installation. Transmit-only LTE devices shall use their KNX Serial Number for LTE runtime communication.

Requirement for the corresponding receiver

- During the teach-in procedure, the KNX Serial Number of the transmit-only partner device shall be stored in the KNX Serial Number Table of the receiver.
- The size of the KNX Serial Number Table in the receiver shall depend on the number of supported transmit-only partners. The size is device specific and depends on the application.

The Link procedure and the distribution of the Serial Number are specified in clause 8.7.

Transmit-only devices cannot acquire a unique Individual Address. Therefore LTE messages of transmit-only devices shall contain the default Individual Address as Source Address.

LTE messages from different transmit-only devices shall be differentiated by:

- the extended LTE Group Address which contains the unique Serial Number and LTE zoning information. Due to the nature of the Serial Number, each extended LTE Group Address is unique and in principle sufficient for unambiguous linking. Therefore LTE zoning information is an additional binding information and may contain the default value; see ”Configuration and usage of LTE zoning information” below
- Datapoint address: Interface Object Type and Property Identifier.

Configuration and usage of LTE zoning information

- Device with local user interface for zone configuration

  The LTE zoning information shall be configurable and used as on bidirectional devices.

- Device without local user interface for zone configuration

  For transmit-only devices, configuration of LTE zoning information can be a problem. These devices may have no local user interface except a push button, and remote configuration of LTE zoning information by a tool is also not possible. For LTE runtime Interworking these devices shall use default LTE zoning information.

  This is, linking with the corresponding receiver(s) is based on the unique KNX Serial Number of the sender only. The effective LTE zoning information may be configured in the receiver that may redistribute the information as a proxy with the proper zoning information in the RF system or in case of a Media Coupler to a TP or PL network.

Spontaneous LTE runtime communication shall be enabled after device configuration and completion of the teach-in procedure.

6.7.3.2 Receive-only devices

Receive-only devices are not allowed.

6.7.4 Redistribution of LTE messages from transmit-only devices in the Domain

If a transmit-only device has a 1:N relation with multiple receivers it is in principle possible to link the sender with N receivers separately. This is, the linking procedure as specified in clause 8.7 shall be executed N times and the KNX Serial Number of the sender shall be memorized in each of the N receivers.

Depending on the application and the number N of receivers it may be more efficient if the transmit-only device were linked with one bidirectional device that shall then redistribute the message like a proxy within the Domain including proper LTE zoning information. The Datapoint address (this is the Interface Object Type Property Identifier) shall not be changed.

If the transmit-only device uses default LTE zoning information the corresponding zoning information within the Domain shall be configured on the proxy.

This mechanism creates more communication traffic but reduces the configuration effort.

This feature is optional and product or application specific.

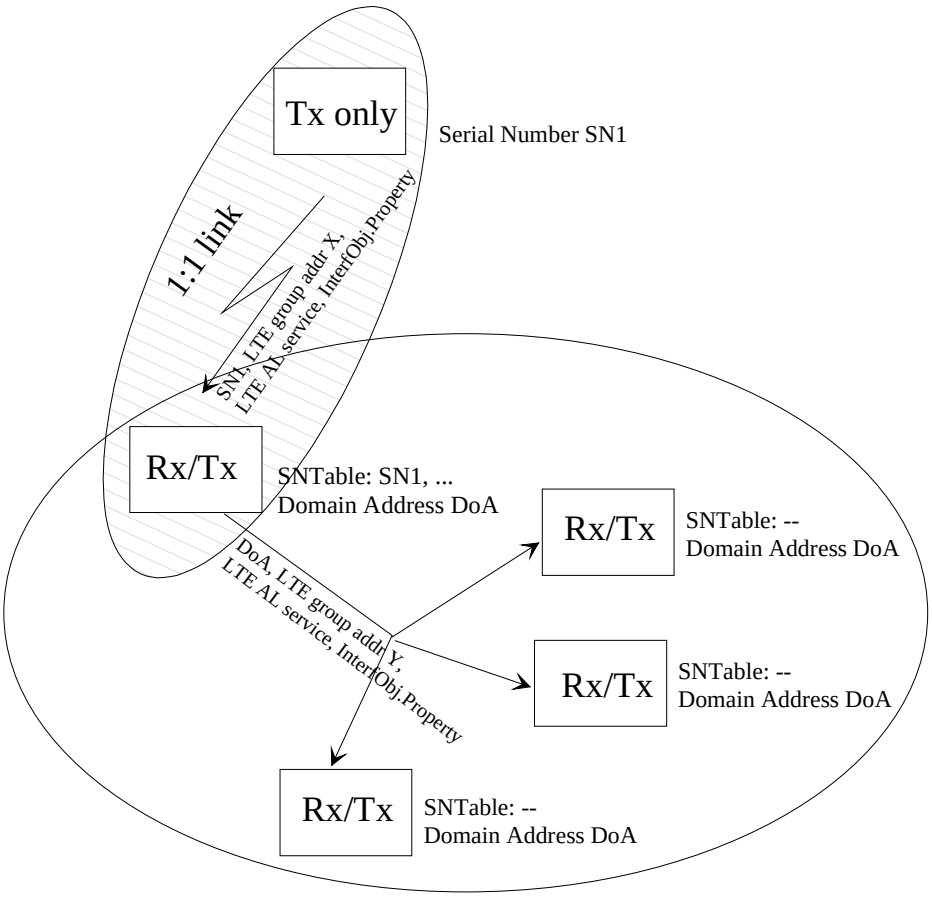


Figure 17 - Proxy mechanism: redistribution of messages within the Domain

NOTE This feature will be used in the synchronous RF BiBat system where the proxy mechanism is implemented in the BiBat master device, see [02].

6.7.5 Mapping of LTE messages from transmit-only devices to wired media

The transmit-only device shall be linked with the “Media Coupler” as described in clause 8.7.

The Media Coupler will redistribute LTE messages from the transmit-only device on the wired medium like a proxy including proper LTE zoning information. If the transmit-only device uses default LTE zoning information, the corresponding LTE zoning information on the wired medium shall be configured on the Media Coupler. The Datapoint address (i.e. Interface Object Type and Property Identifier) shall not be changed.

Since the Media Coupler acts as a proxy for LTE runtime interworking, the transmit-only device is not “visible” as an individual device on TP or PL.

Translation of source Individual Address: according to [02] clause “The Layer-2 of an RF-TP Media Coupler”.

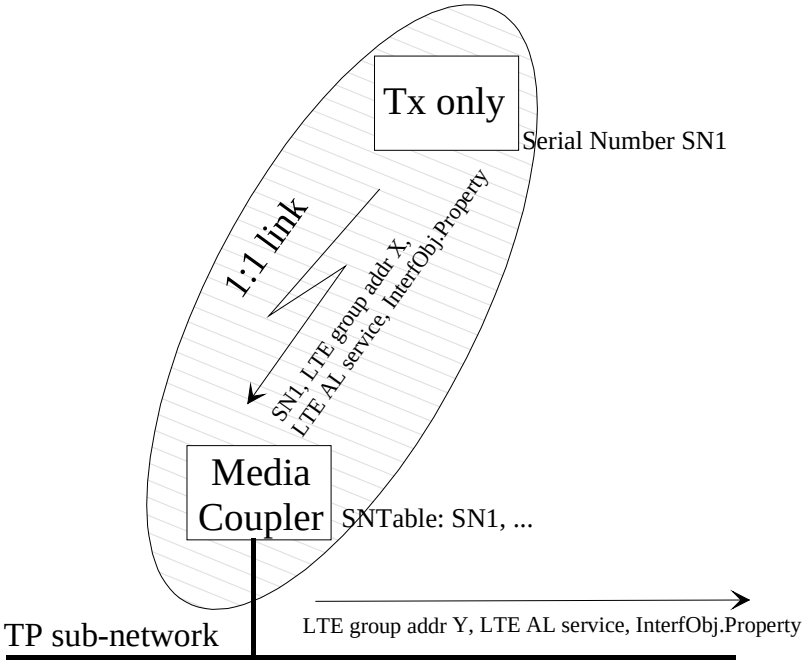


Figure 18 - Mapping of unidirectional devices to TP1 network (proxy)

6.7.6 Mapping of LTE messages from bidirectional devices to wired media

The bidirectional devices shall be linked with the “Media Coupler” via Domain Address as specified in clause 8.8.

LTE messages from RF shall be redistributed on TP or PL medium without change of the LTE zoning information or Datapoint addressing (i.e. Interface Object Type and Property Identifier).

Translation of source Individual Address: according to [02] clause “The Layer-2 of an RF-TP Media Coupler”.

Filtering of LTE messages from RF to TP or PL medium may be incorporated in the Media Coupler (standard LTE filtering mechanism).

6.7.7 Mapping of LTE messages from wired media to RF

The RF devices shall be linked with the “Media Coupler” via the RF Domain Address as described in clause 8.8

LTE messages from TP or PL are redistributed on RF medium with additional Domain Address information but without change of LTE zoning information or Datapoint addressing (i.e. Interface Object Type and Property Identifierà).

The original source Individual Address is kept on RF.

Filtering of LTE messages from TP or PL medium to RF may be incorporated in the Media Coupler (standard LTE filtering mechanism).

6.7.8 LTE Runtime Interworking on RF BiBat synchronous system

6.7.8.1 Static application binding

Within a given LTE zone, BiBat devices will exchange various Datapoint values according to a predefined application standard. Zoning information and Datapoint information correspond to a given, specified and predefined functional purpose (static definition).

Asynchronous LTE RF communication supports n:m links and point-to-point communication, whereas in LTE BiBat communication the BiBat Master is involved in the application binding.

- Any LTE downstream communication from e.g. an asynchronous RF device to a BiBat device shall be handled by the BiBat Master. That means that the BiBat Master shall work like a proxy and shall know the predefined application bindings.
- Asynchronous LTE upstream communication shall be received and handled by the BiBat Master or any other asynchronous Bidirectional device according the standard LTE application bindings.

6.7.8.2 Predefined runtime Interworking

LTE runtime Interworking is mainly based on the producer - consumer principle using multicast group addressing in combination with Interface Objects/Properties. LTE messages are „typed“ and contain semantic information.

- Zoning information shall be mapped statically to predefined LTE Group Addresses. On asynchronous RF communication, the LTE Group Addresses shall be extended by the Domain Address (bidirectional communication) or KNX Serial Number (transmit only devices).

  LTE communication to/from BiBat devices shall always use the Domain Address.
- BiBat Master and BiBat Slaves belong to the same Domain. The Domain Address can be considered as a higher-level LTE zone.
- The BiBat Master shall assign LTE group receive-block(s) to BiBat Slaves belonging to the same LTE zone.

  EXAMPLE 1 All BiBat Slaves in the same "Room" zone shall share the same LTE group receive-block(s).
- Different Datapoints within one LTE group shall be addressed by predefined Properties of Interface Objects. For higher efficiency it is allowed that the BiBat Master sends more than one LTE message in the same receive-block (e.g. different properties) but the addressed group receivers shall be identical. See clause “Telegram length” in “KNX RF Data Link Layer extension for BiBat synchronous system” in [02].

- LTE downstream runtime communication shall be managed by the BiBat Master. The BiBat Master shall either be the origin of downstream data (related Functional Blocks are located in the BiBat Master) or the BiBat Master shall collect data from asynchronous devices and redistribute them in the BiBat System according to the application model and LTE Group Addressing.
- LTE upstream communication is always asynchronous and either directed to the BiBat Master or to asynchronous LTE devices in the same Domain.

EXAMPLE 2

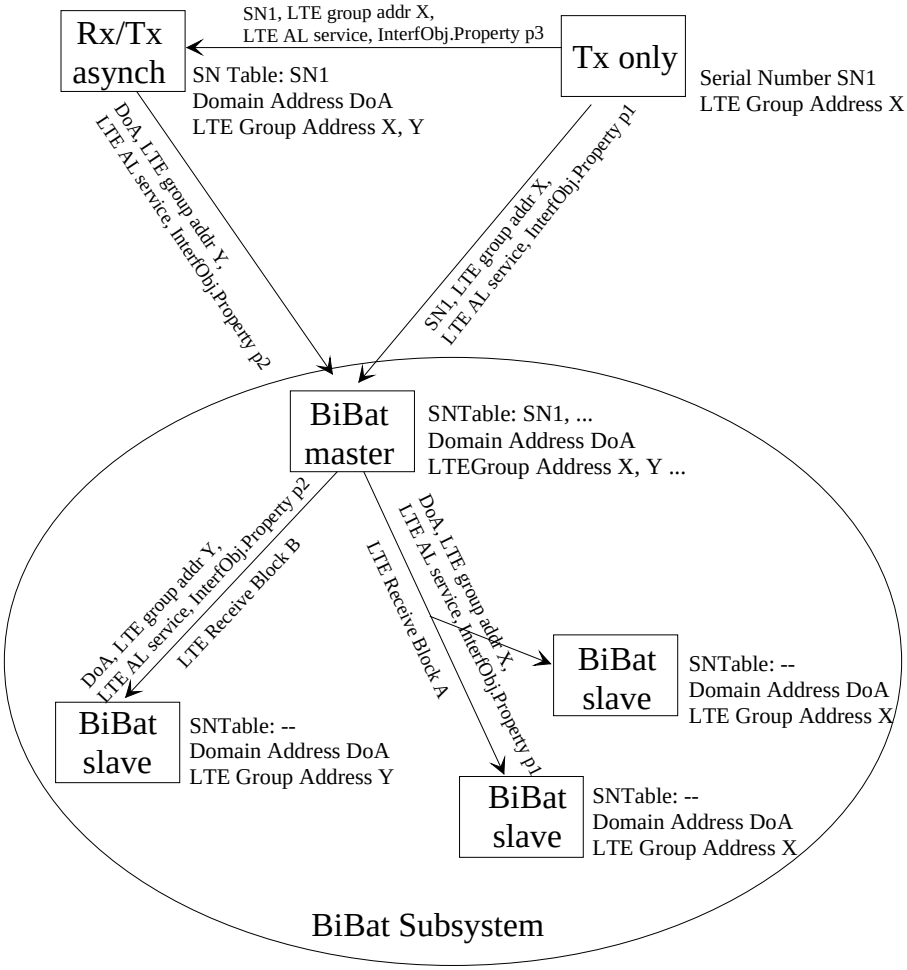


Figure 19: – LTE downstream communication

6.7.8.3 LTE Runtime Interworking

For the specification of the KNX RF BiBat synchronous communication system, please refer to [02].

Downstream runtime Interworking of LTE BiBat devices shall be based on 3 addressing mechanisms:

1. **LTE broadcast addressing**

   The values of various Datapoints can be transmitted in the same block; the Datapoints shall be addressed by the Interface Object Type and Property Identifier.

   Runtime Interworking is mainly based on LTE-InfoReport service.

   ⇒ All LTE BiBat Slaves shall share the same broadcast receive-block(s).

2. **LTE multicast (group) addressing depending on the configured zoning information.**

   It is allowed that different LTE Group Addresses are used in the same receive-block. If e.g. two BiBat Slaves support the same Functional Blocks with the same zones it is possible to share the same receive-block using different LTE Group Addresses.

   The values of various Datapoints can be transmitted in the same block, the Datapoints are addressed by the Interface Object Type and Property Identifier.

   Runtime Interworking is mainly based on LTE-InfoReport service.

   ⇒ All LTE BiBat Slaves with the same zoning information shall share the same group receive-block(s).

3. **Point-to-point addressing of Properties; read- or write service; standard mechanism**

   These services are used for point-to-point communication, e.g. writing of runtime-parameters or if confirmed services are needed for higher reliability of the data exchange.

   ⇒ Receive-blocks can be individually assigned to individual devices in smaller systems.

   ⇒ Due to the limited number of blocks per section it may be necessary in larger systems that blocks have to be shared for Individual Addressing.

Multiple LTE messages in one receive-block:

Depending on the application, the number of devices in a BiBat System, the number of used receive-blocks in a system and the battery capacity of the BiBat Slave it may be necessary to send more than one message within the same block. Up to 3 concatenated telegrams in one block are allowed, see clause 7.3.3 “Telegram length” in [02].

Restrictions:

The same group of BiBat Slaves shall be addressed

⇒ LTE broadcast or group message for BiBat Slaves with the same zoning information

⇒ This shall be restricted to unconfirmed LTE InfoReport-service and LTE Write-service, because this does not trigger responses in the receiver.

⇒ The Property Datapoint length (per telegram) is currently limited to 10 octets.

7 LTE-HEE protocol mechanisms and services

7.1 Usage of the L_Data_Extended group message format for LTE-HEE

If LTE-HEE should be implemented on existing L_Data message format using shared variable addressing mechanisms, much more than the whole 16 bit Group Address range would be needed to encode zoning information and Datapoint addresses.

Therefore LTE-HEE runtime Interworking using group communication is completely based on the L_Data_Extended frame format that supports enough encoding space for zoning information, Datapoint addressing and enables in addition long messages.

The encoding of L_Data_Extended frames is medium-dependent. (The below encoding structure is only informative, please refer to the medium dependent Data Link Layer specifications in [01].)

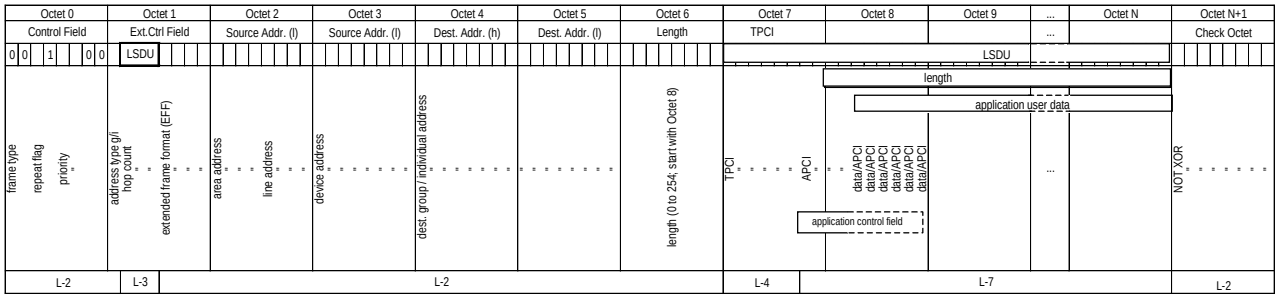


Figure 20 - Example L_Data Request Extended Frame Format on TP1

L_Data-Extended frame format is selected by the Data Link Layer Parameter ‘Frame Format’ which consists of a flag ‘Frame Type’ (FT=standard/extended) and a 4 bit ‘Extended Frame Format’ (EFF) parameter for extended frames. Only informative, see data Link Layer general (extended).

Frame Format parameter

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Frame Type Parameter				Extended Frame Format EFF			
FTP	0	0	0	t	t	t	t
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	**0**	**1**	**x**	**x**

FTP = Frame Type Parameter (0 = Standard, 1 = Extended)
to be mapped medium-dependent to the Frame Type flag FT in the Control Field,
e.g. for FTP=0 (Standard) the corresponding Frame Type FT bit in the control field is 1 on TP1

EFF = Frame Format in case of FTP=1 = Extended Frame Format

Standard Frame Format Standard Group or Individual
Extended Frame Format Standard Group or Individual
LTE-HEE extended address type
All other codes are reserved for future use

Figure 21 - Frame format Parameter

The L_Data_Extended frame format shall be selected by the Data Link Layer parameter ‘Frame Format’ that shall consist of the flag ‘Frame Type Parameter’ (FTP = standard/extended) and a 4 bit ‘Extended Frame Format’ (EFF) parameter for extended frames. For the specification, please refer to [03] clause “Usage of Frame Format”.

For LTE-HEE frames the range EFF= 01xx shall be used. The two bits EFF_1, EFF_0 shall contain a 2 bit extension of the Group Address used in LTE-HEE messages.

7.2 LTE-HEE Group Address extension

The full 16 bit Group Address together with extended frame format range EFF=01xx is used to map LTE-HEE specific zoning information (see clause 5) **statically** to Group Addresses.
This leads to four independent 16 bit Group Address ranges.

EFF_3	**EFF_2**	**EFF_1**	**EFF_0**	**Interpretation of 16 bit DA Group Address**
0	1	0	0	Geographical tags: Apartment/Floor 01...63.R.S
0	1	0	1	Geographical tags: Apartment/Floor 64...126.R.S
0	1	1	0	range of Application specific tags
0	1	1	1	Unassigned (peripheral) tags & broadcast

Figure 22 - Mapping of LTE-HEE logical tag types to EFF field

The interpretation of the 16 bit Group Address field DA depends on the EFF value. There are no conflicts between standard 16 bit Group Addresses and LTE-HEE Group Addresses (full separation).

7.2.1 Mapping of geographical tags

The Group Address range of geographical tags is divided into a section for Apartment/Floor numbers 1...63 and a section for Apartment/Floor numbers 64...126. See Figure 22.

This allows mapping of the full range geographical tags as required in clause 5.1.3

Mapping:

<table>
<tr><th colspan="4">EFF</th><th colspan="16">DA Group Address</th></tr>
<tr><th>b3</th><th>b2</th><th>b1</th><th>b0</th><th>b15
a</th><th>b14
a</th><th>b13
a</th><th>b12
a</th><th>b11
a</th><th>b10
a</th><th>b9
r</th><th>b8
r</th><th>b7
r</th><th>b6
r</th><th>b5
r</th><th>b4
r</th><th>b3
s</th><th>b2
s</th><th>b1
s</th><th>b0
s</th></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td colspan="6">Apartment/Floor 1..63
0 = Wildcard [1)]</td><td colspan="6" rowspan="2">Room 1..63
0= Wildcard</td><td colspan="4" rowspan="2">Subzone 1..15
0 = Wildcard</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td colspan="6">Apartment/Floor 64..126
0 = Wildcard [1)]</td></tr>
</table>

[1)] Wildcard on Apartment/Floor: $DA_{b15..b10}$ all bits set to 0 for both values of EFF_{b0} (no differentiation)

Example: message to zone Apartment= 21, Room=7, Subzone=4 is mapped to the Group Address:

EFF				DA Group Address															
b3	b2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	1	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	1	0	0

Example: message to zone Apartment= 64, Room= *, Subzone= * is mapped to the Group Address:

EFF				DA Group Address															
b3	b2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Example: message to zone Apartment= *, Room= 8, Subzone= * is mapped to the Group Address:

EFF				DA Group Address															
b3	b2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	1	0	x	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

7.2.2 Mapping of application specific tags

The application specific tags must be defined by the Application Interworking Groups.

The available DA Group Address range of 16 Bit (64k) is split up into 16 sections for different Application Domains. Therefore 12 Bits (4k Group Addresses) are reserved for each application domain like e.g. HVAC. In HVAC applications there is currently most need for application specific tags. The remaining 60k Group Addresses are reserved for future use in other application domains.

Management of the 64k available DA Group Addresses for the mapping of these tags is handled by the System Group / TF Interworking.

NOTE Separation of messages between different application domains is also ensured by the application specific different Interface Object Types. But additional message filtering at Link Level is more efficient in terms of CPU load. Dedicated Group Addresses for application specific tags also enable an efficient way to find out and visualise zoning status in a given application.

Mapping:

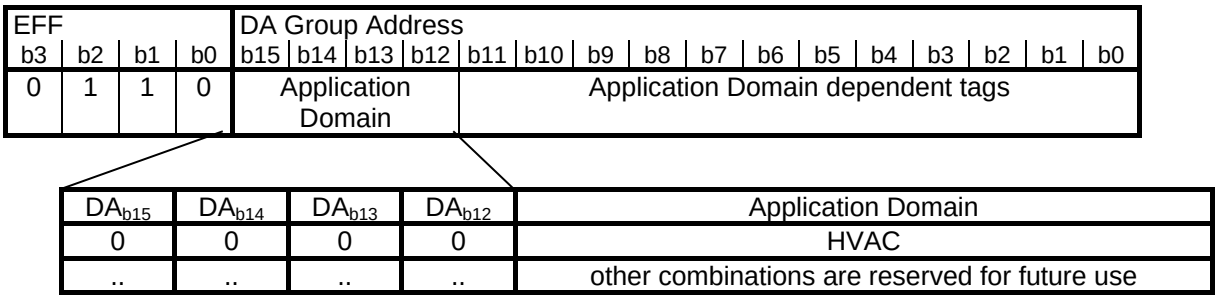


Figure 23 - Application Domain dependent mapping of LTE-HEE application specific tags

7.2.2.1 Application specific tags for HVAC

The following table lists the mapping of HVAC zones to Group Addresses. The list is not final and contains the types of HVAC zones as defined today in the HVAC Application Interworking Standard. The encoding scheme contains sufficiently reserved address space for future extensions.

DA Group Address																
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	All HVAC (broadcast)
				0	0	0	0	0	0	0	..... etc.					reserved
				0	0	0	0	0	0	1	DistrSegm**H** 1..31, 0 = Wildcard					
				0	0	0	0	0	1	0	DistrSegm**C** 1..31, 0 = Wildcard					
				0	0	0	0	0	1	1	DistrSegm**V** 1..31, 0 = Wildcard					
				0	0	0	0	1	0	0	DHWZone 1..31 , 0 = Wildcard					
				0	0	0	0	1	0	1	OutsideSensorZone 1..31 , 0 = Wildcard					
				0	0	0	0	1	1	0	HVACCalendarZone 1..31 , 0 = Wildcard					
				0	0	0	..	..	..	..	..... etc.					reserved
				0	0	1	ProdSegm**H** 1..16				Producer (heating) 1..31 , 0 = Wildcard					
				0	1	0	ProdSegm**C** 1..16				Producer (cooling) 1..31 , 0 = Wildcard					
				0 1	1 X	1 X	..... etc.									reserved

Figure 24 - Mapping of application specific tags for HVAC

Broadcast 'All HVAC'

- One Group Address is reserved to distribute messages which are common within the HVAC application domain.

  IMPORTANT:

  - The usage of the HVAC broadcast together with LTE InfoReport or Write service is specified at Datapoint level in the Application Specifications and restricted to a defined set of Datapoints. In HVAC devices a received HVAC broadcast message will be accepted at Data Link Layer level but the Datapoint is NOT a priori accepted at the level of Application.
  - LTE Read request using the HVAC broadcast address is also restricted to a defined set of Datapoints according to the HVAC Application Specifications. For only those Datapoints the corresponding Response shall be generated in the receivers.
  - In both cases the proper usage of the HVAC broadcast in combination with a Datapoint is checked in the receiving Application Interface Layer.

Mapping of 'DistributionSegment' for Hot Water, Cold Water and Ventilation

- For hot water, cold water and air distribution LTE supports 3 x 31 independent Distribution Segments
- Wildcard addressing for each class (H, C, V) of DistributionSegment is supported.

Mapping of 'DHWZone'

- LTE supports up to 31 independent DHWZones for domestic hot water zoning (independent of the building structure)
- Wildcard addressing for DHWZone is supported.

Mapping of 'OutsideSensorZone'

- LTE supports up to 31 independent zones for outside sensors (outside temperature, wind speed, sun intensity). In practice up to eight zones will usually be sufficient
- Wildcard addressing for OutsideSensorZone is supported.

Mapping of 'HVACCalendarZone'

- A 'Calendar' provides information about exceptions in time dependent operating of a plant or part of a plant. Calendar information like holiday-period is mainly used in commercial buildings and is often related to several building locations, not only to one room or floor/apartment. Therefore calendar information is not distributed in geographical zones like apartment_x.room_y, but distributed in a 'HVAC Calendar Zone'
- LTE supports up to 31 independent calendar zones
- Wildcard addressing is supported.

Mapping of 'ProdSegm.Producer' for Hot Water and Cold Water

- LTE supports up to 31 independent Producers (e.g. boilers) per ProdSegm (e.g. boiler cascade).
- Up to 16 different and independent ProdSegmH for Hot Water are supported
- Up to 16 different and independent ProdSegmC for Cold Water are supported
- Wildcard addressing for all Producers in a ProdSegm is supported.
  Wildcard addressing for all ProdSegm is not foreseen (no application need).

More details concerning HVAC zones are described in the HVAC Application Interworking Specifications

7.2.3 Mapping of unassigned (peripheral) tags

The Group Address range of unassigned tags is divided into two sections.

- one section (4k) for manually configured tags
- one section (60k) reserved for future usage

Mapping

EFF				DA Group Address															
b3	b2	b1	b0	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	configurable tags (b_{11} .... b_0 >0)											
				0	0	0	1	reserved for future usage											
				.....															
				1	1	1	1												

Broadcast

Figure 25 - Mapping of unassigned tags

7.2.3.1 LTE-HEE broadcast

One address EFF=0111 and DA = 0000h is reserved for broadcast messages. This address is not configurable by selecting a logical tag.

Broadcast message to “all LTE-HEE” could be used for some specific applications like e.g. distribution of alarm messages, distribution of system clock information etc.

IMPORTANT:

- The usage of the LTE-HEE broadcast together with LTE InfoReport or Write service is specified at Datapoint level in the Application Specifications and restricted to a defined set of Datapoints. In LTE devices a received LTE-HEE broadcast message will be accepted at Data Link Layer level but the Datapoint is NOT a priori accepted at the level of Application.
- LTE Read request using the LTE-HEE broadcast address is also restricted to a defined set of Datapoints according to the Application Specifications. For only those Datapoints the corresponding Response shall be generated in the receivers.
- In both cases the proper usage of the LTE-HEE broadcast in combination with a Datapoint is checked in the receiving Application Interface Layer.

7.2.3.2 Configurable tags

The Group Address range for these tags is flat (no structuring, nesting etc.). It is up to the manufacturer of a device if the whole 4k address range is made available for configuration of the tag. If only a sub-set is configurable (e.g. the lower 8 bits) the remaining bits must be set to 0.

7.3 LTE-HEE Group Address tables

7.3.1 Address table structure

For LTE-HEE message handling in the **Receiver,** the Data Link Layer uses specific Group Address tables which support Wildcard / Sniffer mechanisms.

These Group Address tables are generated out of the set of "Zone List" of each Functional Block as described in clause 6.3.6. The LTE Group Address tables are used in the receiver (Link Layer) to accept or reject received LTE group messages.

LTE Group Address tables are implemented as Properties of max. 4 Group Address Table Interface Objects. The following clause is only informative; please refer to the Resource definitions of System 300 and other Profiles that support LTE in [06].

Property Name	Property ID	Type	Optional / Mandatory / Writable	
Object Type	PID_OBJECT_TYPE	PDT_UNSIGNED_INT	M	Addressable Object 1
Object Name	PID_OBJECT_NAME	PDT_UNSIGNED_CHAR[ ]	O	Name of the Address table
Load Control	PID_LOAD_STATE_CONTROL	PDT_CONTROL	M / W	for further Information see Load / State machines
Address table Pointer	PID_TABLE_Reference	PDT_GENERIC_03	O	Pointer to Address table and maximum entries
Frame Format	PID_EXT_FRAMEFORMAT	PDT_UNSIGNED_CHAR	O	Extended Frame Format 0001b – 1111b
Address Table Format 0	PID_TABLE (23)	PDT_UNSIGNED_INT [ ]	M / W	Address table
Address Table Format 1	PID_ADDRTAB1 (52)	PDT_UNSIGNED_INT[]	(O)	
..				

Figure 26 - Group Address Table Interface Object

For each class of LTE logical tags (depending on EFF field) one extended Group Address Table Interface Object exists with the 'Address Table Format 1' Property. Therefore for LTE max. 4 Group Address Table Interface Objects are possible. Each Address Table consists of a set of 0..N address entries according to the number of supported zone addresses.

The 'Frame Format' Property includes the value of the Frame type and the Extended Frame Format (EFF) for which the Group Address Table Interface Object and 'Address Table Format 1' Property is valid.

'Frame Format'				**Group Address Table Interface Objects: Property 'Address Table Format 1'**			
EFF_3	**EFF_2**	**EFF_1**	**EFF_0**				
0	1	**0**	**0**	**Object for Geographical tags: Apartment/Floor 01...63.R.S**			
				Address entry 00_1	Address entry 00_2	……	Address entry 00_a
0	1	**0**	**1**	**Object for Geographical tags: Apartment/Floor 64...126.R.S**			
				Address entry 01_1	Address entry 01_2	……	Address entry 01_b
0	1	**1**	**0**	**Object for Application specific tags**			
				Address entry 10_1	Address entry 10_2	……	Address entry 10_c
0	1	**1**	**1**	**Object for Unassigned (peripheral) tags**			
				Address entry 11_1	Address entry 11_2	……	Address entry 11_d

Figure 27 - Set of Group Address Table Interface Objects for each LTE 'Frame Format'

The 'Address Table Format 1' is an array of 0..N entries. Each entry represents one single Group Address or a set of Group Addresses supporting Wildcard/Sniffer functionality.

Each entry consists of two unsigned integer. The first integer includes the base address and the second a mask with the valid bits of the base address. The mask is used to encode efficiently a range of Group Addresses for Sniffer functionality instead of listing each address individually.

The Rule for address matching in the receiver is :

If (Group Address AND mask) is equal to (base address AND mask) then match

Theoretical Address Table Example 1:

Base Address	Mask	Matching addresses
1234h	FFFFh	1234h
1230h	FFFFh	1230h
1200h	FFF0h	1200h – 120Fh (Sniffer)

Theoretical Address Table Example 2:

Base Address	Mask	Matching addresses
4805h	FF0Fh	48x5h x = 0…F (Sniffer)
4800h	FF00h	4800h – 48FFh (Sniffer)

The content of the LTE Group Address tables depends on the application of a device (activated Functional Blocks) and the selected zoning information. LTE Group Address tables are therefore calculated at configuration/run-time by the application program as a parameter of the data link layer.

7.3.2 HVAC example

A rather complex device called “Apartment Manager” contains Functional Blocks for:

- Individual room management (e.g. link to individual room controllers in the same apartment); configured **Apartment-No = 14**
  The Apartment Manager is also a Sniffer for all room & subzone information within apartment 14
- Heat demand calculation for the entire apartment (link to pre-controller in corresponding DistrSegmH);
  configured **DistrSegmH-No = 3**. No Sniffer functionality for other DistrSegmH
- Domestic hot water control management (e.g. link to corresponding domestic hot water controller); configured **DHWZone-No = 7**. No Sniffer functionality for other DHW zones
- The apartment manager uses the outside temperature from the configured **OutsideSensorZone-No = 1** but shall be able to take the temperature of another outside sensor zone in case of failure of the sensor in the configured zone (Sniffer).
- For “Zone List” generation see clause 6.3.6

LTE Group Address Table for EFF=00 (geographical):

Base Address																**Mask**															
b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Apartment/Floor						Room						Subzone				Apartment/Floor						Room						Subzone			
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The table contains 3 entries:

- A.R.S = 14.*.* (Wildcard address 14.*.*)
- A.R.S = 14.x.x (Sniffer on 14.R.S)
- A.R.S = *.*.* (Wildcard address on all building location)
  used for Datapoints supporting the AL services InfoReport.ind, Write.ind, Read.ind and Response.ind

LTE Group Address Table for EFF= 10 (application specific):

Base Address																**Mask**															
b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
fixed for DistrSegmH											DistrSegmH					fixed for DistrSegmH											DistrSegmH				
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
fixed for DHWZone											DHWZone					fixed for DHWZone											DHWZone				
0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
fixed for OutsideSensor Zone											OutsideSensor Zone					fixed for OutsideSensor Zone											OutsideSensor Zone				
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

The table contains totally 6 entries:

- 2 entries for DistrSegmH zone:
  - DistrSegmH = 3
  - DistrSegmH = * (Wildcard)
    used for Datapoints supporting the AL services Write.ind, Read.ind, InfoReport.ind and Response.ind
- 2 entries for DHWZone:
  - DHWZone = 7
  - DHWZone = * (Wildcard)
    used for Datapoints supporting the AL services Write.ind, Read.ind, InfoReport.ind and Response.ind
- 2 entries for OutsideSensorZone:
  - OutsideSensorZone = 1
  - Sniffer on all OutsideSensorZones
    used for reception of Datapoint ‘TempOutside’ supporting the AL service InfoReport.ind, Response.ind

The example shows that even for a complex device the size of the Group Address tables for LTE-HEE messages is rather small because of the Wildcard and Sniffer features.

7.4 Layer 2 Acknowledgement of LTE-HEE messages

7.4.1 Normal conditions

L_Data.req and L_Data.ind service primitives have the parameter ack_request also on extended frame format as specified in [03].

Mechanism on physical medium TP1

- On TP1 Layer-2 acknowledge is supported also for multicast messages. The receiver of a properly received LTE L_Data-Extended message will generate a Layer-2 acknowledge if the LTE Group Address is contained in the LTE Group Address Table (similar procedure as for standard group messages).
- LTE implementations with systematic Layer-2 acknowledge generation by the receiver, regardless of the LTE Group Address, are however allowed. In such implementation every properly received LTE L_Data-Extended message will be acknowledged.

  NOTE Standard messages are by the same devices acknowledged as usual.
  These can be ACK, NACK and BUSY. The only difference is that the LTE destination address is not evaluated.
  The product documentation shall specify this.
- Frame repetitions for LTE L_Data-Extended messages in case of acknowledge time-out, NAK or BUSY are not always appropriate and shall be disabled depending on the LTE Application Layer service.
  The behaviour is controlled by the parameter ack_request in the LTE Application Layer service primitive:
  - ack_request = 'requested' ⇒ frame repetitions are enabled
  - ack_request = 'don't care' ⇒ no frame repetitions
- Frame repetitions are not appropriate for the **A_GroupPropValue_InfoReport** service since the sender of the message does not know if there is at least one device consuming the message. In a producer/consumer system it may happen that a device is distributing data and no device is consuming it. In this case the parameter ack_request set to 'requested' would lead to unnecessary frame repetitions and traffic.
  In the parameter list of A_GroupPropValue_InfoReport.req service primitive the parameter ack_request is therefore not available.
  The Application Layer shall set the ack_request parameter in the corresponding T_Data_Tag_Group.req service primitive automatically to 'don't care'.
- For the **A_GroupPropValue_Read.req**, **A_GroupPropValue_Response.req** and **A_GroupPropValue_Write.req** service primitives the ack_request parameter is meaningful and Layer-2 acknowledge mechanism with frame repetitions can be selected by the application.
- LTE implementations having the ack_request parameter generally set to 0 = “don't care” are allowed and LTE frames are in this case never repeated.

Mechanism on physical medium PL110

- On the PL110 medium Layer-2 acknowledge on standard group messages is possible by setting a unique group response flag to each assigned Group Address (one group "speaker"). This concept is not adapted to LTE group communication.
- For LTE messages using multicast addressing, Layer-2 acknowledge and frame repetitions are therefore not applicable.
- The ack_request parameter of the LTE Application Layer service primitives is on these media a dummy parameter and ignored by the lower layers. This is, the ack_request parameter is mapped per default to 'don't care' in the corresponding L_Data service primitive.

7.4.2 Error and exception handling

Only useful combinations of the ack_request parameter / LTE Application Layer service primitive / physical medium shall be accepted by the LTE protocol stack implementation. Otherwise the ack_request parameter shall be mapped per default to 'don't care'.

7.5 Transport Layer control information for LTE-HEE messages (informative)

An own TPCI for LTE-HEE messages (Transport Layer Service) is used for better separation of message handling in the protocol stack of standard messages and LTE-HEE messages (translation of Group Addresses into/from Interworking references). This clause is only informative, please refer to [04].for the Transport Layer requirements.

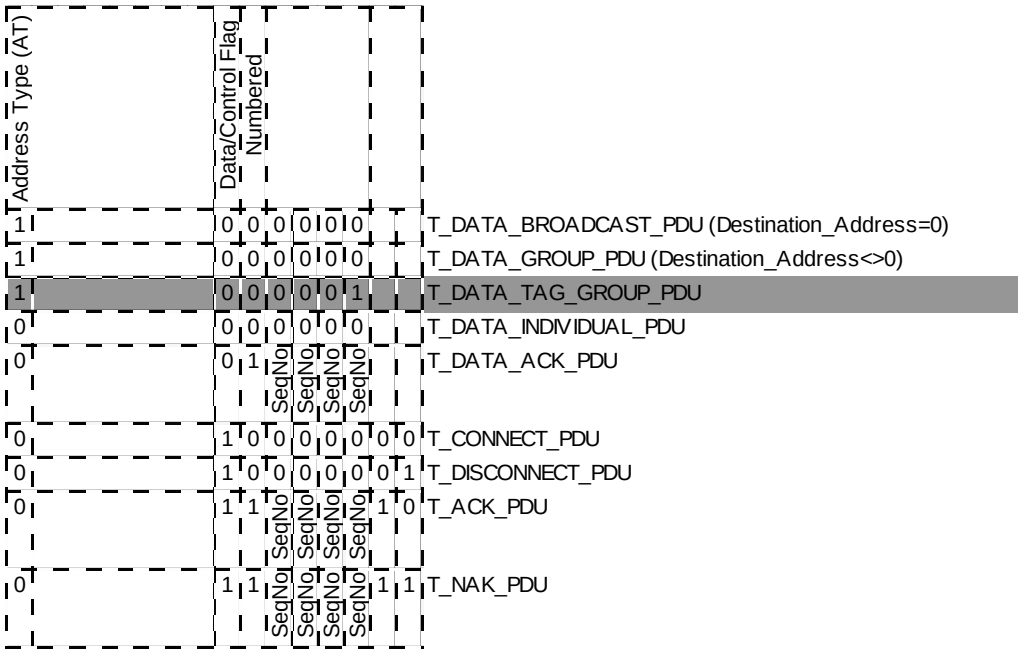


Figure 28 - TPCI for LTE-HEE messages: T_DATA_TAG_GROUP_PDU

7.6 Application Layer services for LTE-HEE messages

7.6.1 Usage of Interface Objects for LTE-HEE runtime Interworking

The concept of Functional Blocks used for application modelling is mapped to Interface Object structures on the level of communication protocol.

I.e. for every type of Functional Block a corresponding Interface Object Type is assigned statically.

Datapoints of a Functional Block are mapped to Properties of the corresponding Interface Object.

The Interface Object Type together with the Property ID is unique Datapoint address information. System-wide 65 535 Interface Object Types each with 256 different Properties can be addressed.

The Datapoints accessible via LTE-HEE mechanisms for a certain application are defined by the Application Interworking Groups. Mapping of the Functional Blocks and Datapoints to the corresponding Interface Object Type / Property ID list is managed by the KNX Association (global list).

Access mechanism to Interface Objects: individual addressing vs. group addressing

In standard **individual addressing mode** Interface Objects are addressed by the Object Index which is local to the device. Properties within an Interface Object are organised as an array

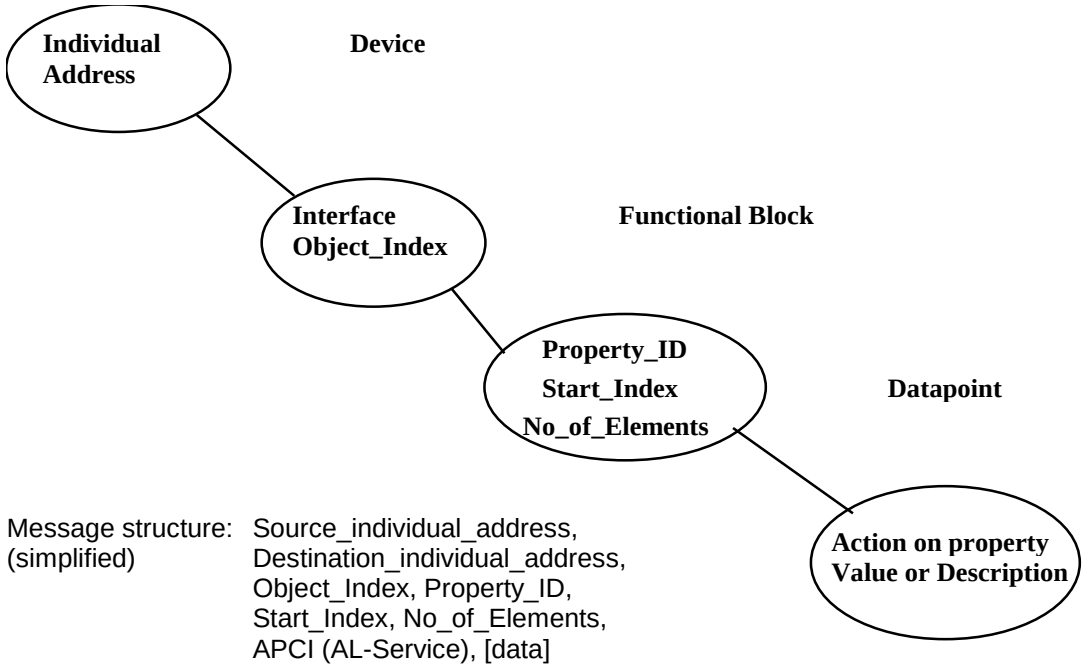


Figure 29 - Access to Interface Objects/Properties using individual addressing

For **LTE-HEE runtime-Interworking using group addressing** usage of the device dependent local Object Index is not appropriate since the Object Index for a given Functional Block is different in every device.

- Interface Objects are addressed by their Interface Object Type and a Instance number
- A device may contain multiple Functional Blocks of the same type. Therefore on the level of protocol a Instance number is needed besides the Interface Object Type for Interface Object addressing.
- The instance number is the **local Object Instance** in a device (and not an instance within a group / zone)
- In practical applications, Datapoints used for LTE-HEE runtime Interworking are not organised as arrays. Therefore a property must not be addressed as an array (only one single element)
  ⇒ no property start index and number of elements in address information
- Only the Property value can be accessed but not the Property description

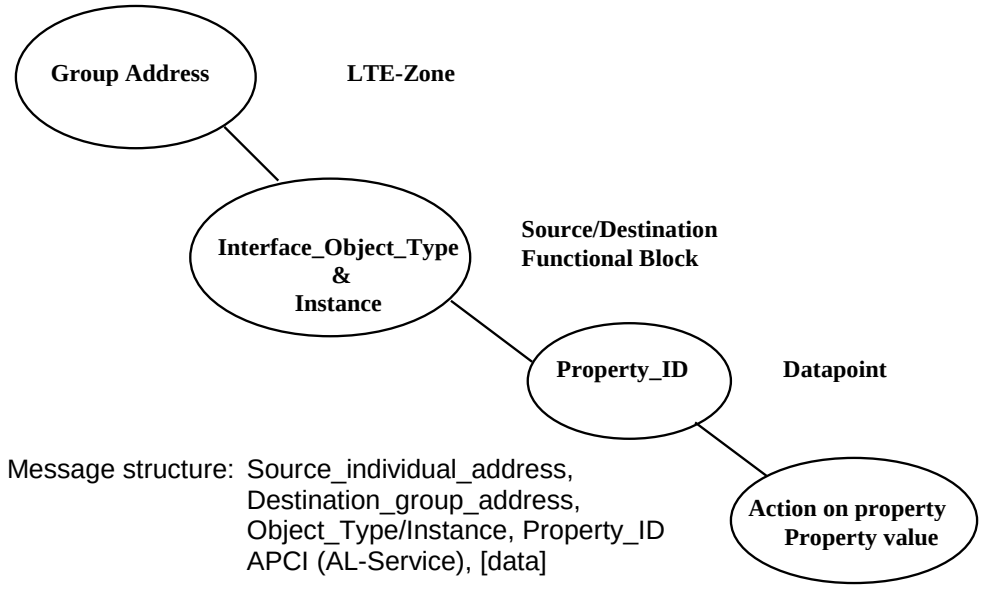


Figure 30 - Access to Interface Objects/Properties using LTE-HEE group addressing

7.6.2 LTE-HEE AL services overview

The following Application Layer Services (APCI) are defined for LTE-HEE runtime Interworking:

- A_GroupPropValue_Read & A_GroupPropValue_Response
- A_GroupPropValue_Write
- A_GroupPropValue_InfoReport

The main difference between Write and InfoReport service and the usage of the services (from an application point of view) is described in clause 6.3.

7.6.3 APDU

The APDU is shown in the following figure (example: octet N° for TP1)

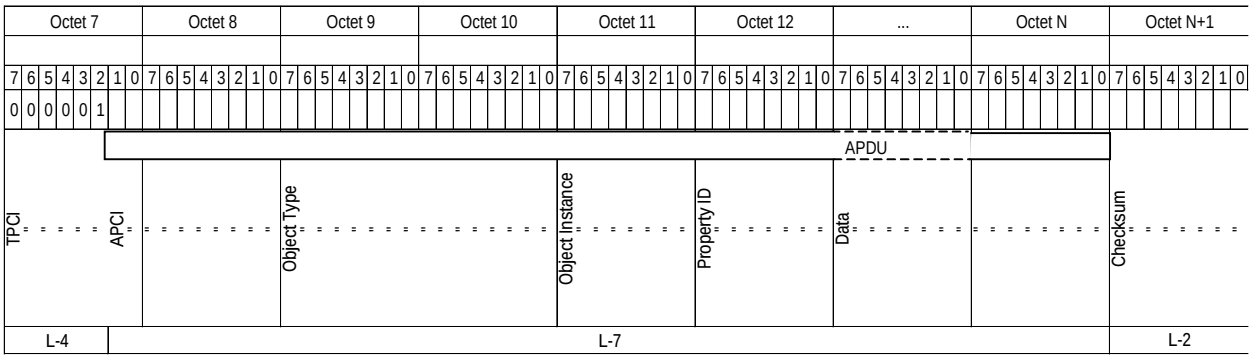


Figure 31 - Format of the LTE-HEE APDU

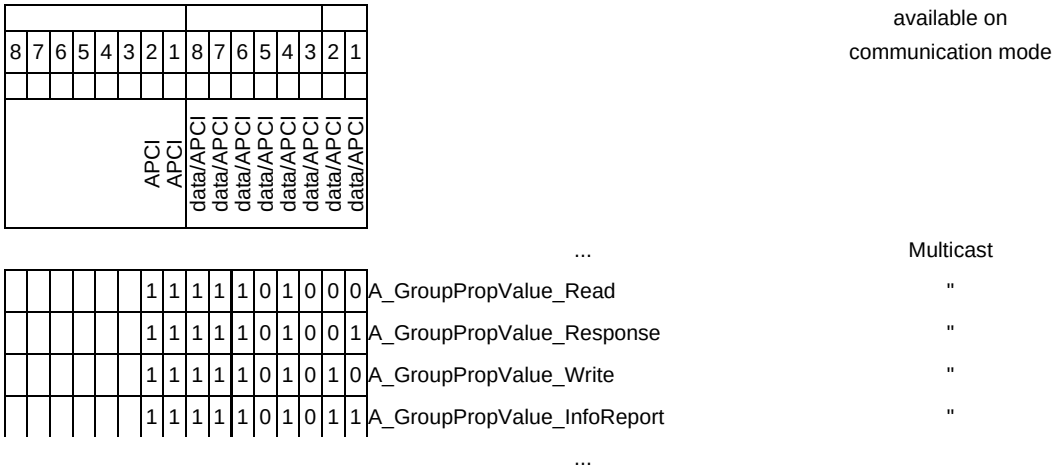
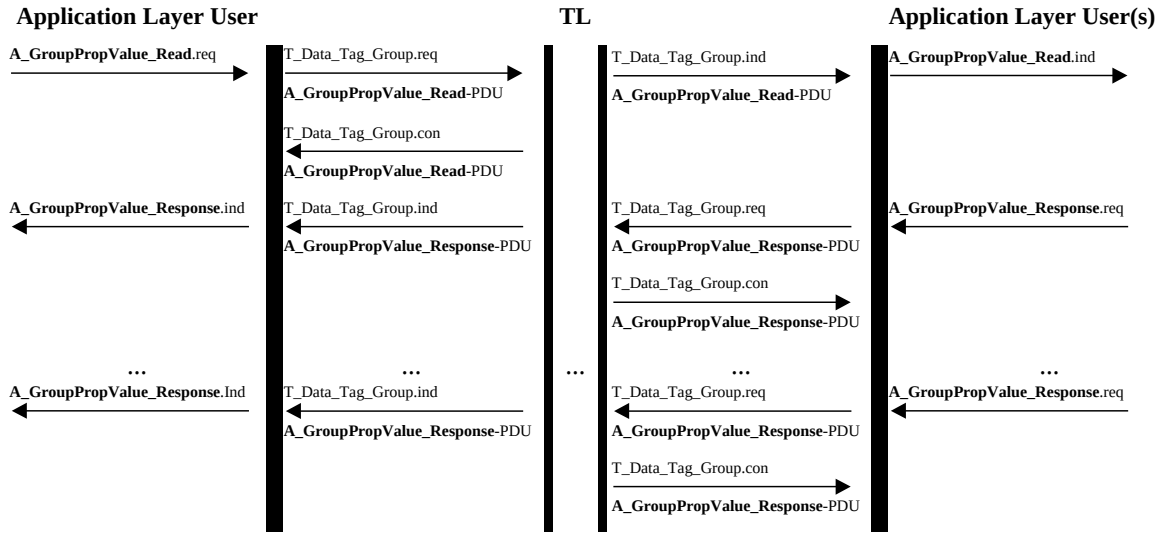


Figure 32 - APCI for LTE-HEE A_GroupPropValue_... services

7.6.4 A_GroupPropValue_Read / Response



The A_GroupPropValue_Read.req primitive is applied by the user of Application Layer, to receive an update of the value(s) of its Application Layer Service Access Points (ASAP) by making none, one or N communication partner(s) respond with an A_GroupPropValue_Response.req.

The local Application Layer accepts the service request, maps the ASAP to the TSAP and passes it with a T_Data_Tag_Group.req to the local Transport Layer.

When one device sends an A_GroupPropValue_Read.req each device which is member of this “group” receives the A_GroupPropValue_Read.ind. “Group” means on single Group Address or a set of Group Addresses because of Wildcard addressing.

The remote Application Layer maps a T_Data_Tag_Group.ind primitive with TSDU= A_GroupPropValue_Read-PDU to an A_GroupPropValue_Read.ind primitive.

After reception of A_GroupPropValue_Read.ind the existence and access to the addressed InterfaceObject.Property is checked by the application program and the application program decides about generation of a Response. The application program may respond to the A_GroupPropValue_Read.ind primitive with an A_GroupPropValue_Response.req primitive containing the value of the addressed Property. No Response is generated if access to the addressed property is not supported.

In one device the remote application program may generate multiple responses if:

- multiple instances of the same Interface Object Type are belonging to different groups which are accessed by group Wildcard mechanism
- multiple instances of the same Interface Object Type belong to the same group

The remote Application Layer accepts the service response, and a T_Data_Tag_Group.req is passed to the local Transport Layer.

The local Application Layer of the service requestor maps a T_Data_Tag_Group.ind primitive with TSDU= A_GroupPropValue_Response-PDU to an A_GroupPropValue_Response.ind primitive.

None, one or more than one A_GroupPropValue_Response.Ind primitive may occur depending on the number of group members that belong to the same group and provide the requested Property.

The procedure shows that A_GroupPropValue service is an **unconfirmed service** because the number of responses is not known by the service requestor. After processing of A_GroupPropValue_Read.req the Application Layer has no “memory” and state machine for this service. And therefore there is no timeout checking for responses in the AL (but probably in the application process if necessary).

Since the service is unconfirmed, there is also no local confirmation as e.g.
A_GroupPropValue_Read.Lcon

Octet 7								Octet 8								Octet 9								Octet 10								Octet 11								Octet 12							
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
TPCI						APCI										Object Type																Object Instance								Property ID							
0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

Figure 33 - A_GroupPropValue_Read_PDU (octet-No example for TP1)

Octet 7								Octet 8								Octet 9								Octet 10								Octet 11								Octet 12								Octet 13...N							
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								
TPCI						APCI										Object Type																Object Instance								Property ID								Data							
0	0	0	0	0	1	1	1	1	1	0	1	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								

Figure 34 - A_GroupPropValue_Response_PDU (octet-No example for TP1)

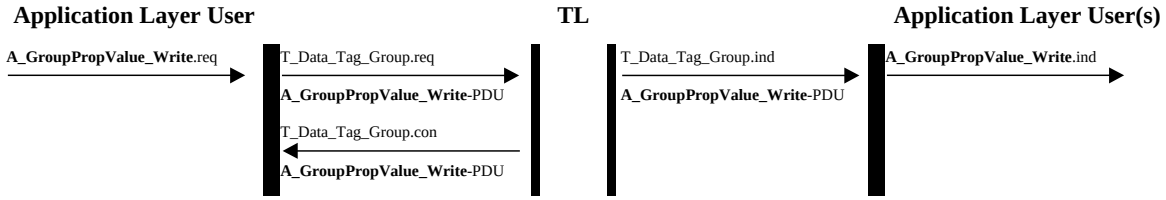
A_GroupPropValue_Read.req	(zone, object_type, object_instance, property_ID, ack_req, priority, hop_count_type)
zone:	zone parameter contains: - EFF → type of LTE-HEE zoning information - Group Address with the mapped zoning information. Zone Wildcard addressing is allowed.
object_type:	Destination-type of the addressed Interface Object
object_instance	Instance of the object to identify more than one object of the same type in one device. For multicast Read service the instance number is not known and not meaningful (dummy field) ⇒ fixed to 0 which means all instances (Wildcard)
property_ID:	the Property-ID of the property of the addressed object
ack_request:	Layer 2 acknowledge and frame repetitions: requested or don't care. - requested: appropriate on TP1 only, enables frame repetitions - don't care: default value for all other media, no frame repetitions
hop_count_type:	hop count 7 or standard
priority:	system, urgent, normal or low priority

A_GroupPropValue_Read.ind	(source_address, zone, object_type, object_instance, property_ID, priority, hop_count_type)
source_address:	Individual address of the sending device
zone:	Same value as in Read.req
object_type:	Same value as in Read.req
object_instance	Same value as in Read.req
property_ID:	Same value as in Read.req
hop_count_type:	Same value as in Read.req
priority:	Same value as in Read.req

A_GroupPropValue_Response.req	(zone, object_type, object_instance, property_ID, data, ack_req, priority, hop_count_type)
zone:	zone parameter contains: - EFF -> type of LTE-HEE zoning information - Group Address with the mapped zoning information. Zone Wildcard addressing is allowed: but the zone information has to be as precise as possible. Zone in the Response.req may be more explicit than zone in the Read.ind Example: zone in Read.ind = A.*.* and in Response.req= A.R.S
object_type:	Source-type of the addressed Interface Object Same value as in Read.ind
object_instance	Instance of the object to identify more than one object of the same type in one device. Must contain explicit instance number (≠0) of the addressed Interface Object (at device level)
property_ID:	the Property-ID of the property of the addressed Interface Object Same value as in Read.ind
data	property value
ack_request:	Layer 2 acknowledge and frame repetitions: requested or don't care. - requested: appropriate on TP1 only, enables frame repetitions - don't care: default value for all other media, no frame repetitions
hop_count_type:	hop count 7 or standard Same value as in Read.ind
priority:	system, urgent, normal or low priority Same value as in Read.ind

A_GroupPropValue_Response.ind	(source_address, zone, object_type, object_instance, property_ID, data, *priority*, *hop_count_type*)
source_address:	Individual address of the sending device
zone:	Same value as in Response.req
object_type:	Same value as in Response.req
object_instance	Same value as in Response.req
property_ID:	Same value as in Response.req
data	Same value as in Response.req
(hop_count_type):	dummy parameter
(priority):	dummy parameter

7.6.5 A_GroupPropValue_Write



The A_GroupPropValue_Write.req primitive is applied by the user of the Application Layer, to write the value of its Application Layer Service Access Point (ASAP) to all connected ASAPs.

The local Application Layer accepts the service request, maps the ASAP to the TSAP and passes it with a T_Data_Tag_Group.req to the local Transport Layer.

When one device sends an A_GroupPropValue_Write.req each device which is member of this “group” receives the A_GroupPropValue_Write.ind. “Group” means on single Group Address or a set of Group Addresses because of Wildcard addressing.

The remote Application Layer maps a T_Data_Tag_Group.ind primitive with TSDU= A_GroupPropValue_Write-PDU to an A_GroupPropValue_Write.ind primitive.

After reception of A_GroupPropValue_Write.ind the existence and access-rights to the addressed InterfaceObject.Property is checked by the application program. The application program ignores the write access if the property does not exist or if the property is read-only. Otherwise the data value is written to the corresponding property.

The service is not confirmed by the remote application process.

There is also no confirmation (of transmission) by e.g. a local A_GroupPropValue_Write.Lcon

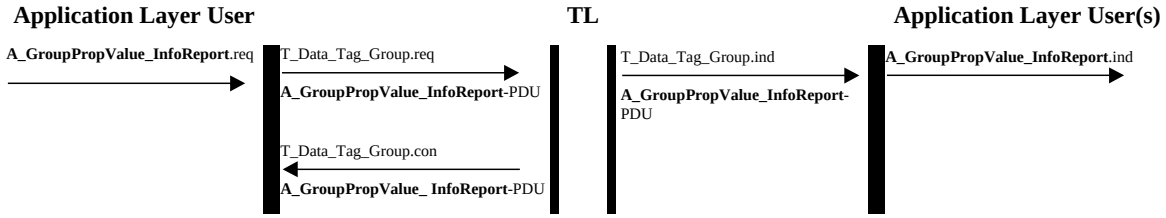
Octet 7								Octet 8								Octet 9								Octet 10								Octet 11								Octet 12								Octet 13...N							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
TPCI								APCI								Object Type																Object Instance								Property ID								Data							
0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x								

Figure 35 - A_GroupPropValue_Write_PDU (octet-No example for TP1)

A_GroupPropValue_Write.req	(zone, object_type, object_instance, property_ID, data, ack_req, priority, hop_count_type)
zone:	zone parameter contains: - EFF -> type of LTE-HEE zoning information - Group Address with the mapped zoning information. Zone Wildcard addressing is allowed
object_type:	Destination-type of the addressed Interface Object
object_instance	Instance of the object to identify more than one object of the same type in one device. For multicast Write service the instance number is not known and not meaningful (dummy field) ⇒ fixed to 0 which means all instances (Wildcard)
property_ID:	the Property-ID of the property of the addressed object
data	written property value
ack_request:	Layer 2 acknowledge and frame repetitions: requested or don't care. - requested: appropriate on TP1 only, enables frame repetitions - don't care: default value for all other media, no frame repetitions
hop_count_type:	hop count 7 or standard
priority:	system, urgent, normal or low priority

A_GroupPropValue_Write.ind	(source_address, zone, object_type, object_instance, property_ID, data, *priority*, *hop_count_type*)
source_address:	Individual address of the sending device
zone:	Same value as in Write.req
object_type:	Same value as in Write.req
object_instance	Same value as in Write.req
property_ID:	Same value as in Write.req
data	Same value as in Write.req
(hop_count_type):	dummy parameter
(priority):	dummy parameter

7.6.6 A_GroupPropValue_InfoReport



The A_GroupPropValue_InfoReport.req primitive is applied by the user of the Application Layer, to distribute the value of its Application Layer Service Access Point (ASAP) to all connected ASAPs.

The local Application Layer accepts the service request, maps the ASAP to the TSAP and passes it with a T_Data_Tag_Group.req to the local Transport Layer.

When one device sends an A_GroupPropValue_InfoReport.req each device which is member of this "group" receives the A_GroupPropValue_InfoReport.ind. "Group" means on single Group Address or a set of Group Addresses because of Wildcard addressing. Receivers may also be "Sniffers" on a set of Group Addresses.

The remote Application Layer maps a T_Data_Tag_Group.ind primitive with TSDU= A_GroupPropValue_InfoReport-PDU to an A_GroupValue_InfoReport.ind primitive.

After reception of A_GroupPropValue_InfoReport.ind the existence of the addressed InterfaceObject.Property is checked by the application program. The application program ignores the data if the device is not a consumer of the property.

Otherwise the data value is accepted but further processing of the data is completely application dependent. Handling of the received data is completely different compared to Write service

- the data may be stored in a database (original or modified value),
- it may be used for further calculations,
- it may be used to trigger a local action
- it may be thrown away because data value is not relevant in the current situation
- etc.

The service is not confirmed by the remote application process.

There is also no confirmation (of transmission) by e.g. a local ***A_GroupPropValue_InfoReport.Lcon***

Octet 7								Octet 8								Octet 9								Octet 10								Octet 11								Octet 12								Octet 13...N							
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								
TPCI						APCI										Object Type																Object Instance								Property ID								Data							
0	0	0	0	0	1	1	1	1	1	1	0	1	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								

Figure 36 - A_GroupPropValue_InfoReport_PDU (octet-No example for TP1)

A_GroupPropValue_InfoReport.req	(zone, object_type, object_instance, property_ID, data, priority, hop_count_type)
zone:	zone parameter contains: - EFF -> type of LTE-HEE zoning information - Group Address with the mapped zoning information. Zone Wildcard addressing is allowed
object_type:	Source-type of the addressed Interworking Object
object_instance	Instance of the object to identify more than one object of the same type in one device. Must contain explicit instance number (≠0) of the addressed Interface Object (at device level)
property_ID:	the Property-ID of the property of the addressed Interface Object
data	property value
hop_count_type:	hop count 7 or standard
priority:	system, urgent, normal or low priority

Note: Frame repetitions are undesirable for the A_GroupPropValue_InfoReport-service and the ack_request parameter, if present, would be fixed to 'don't care' value. The Application Layer shall set the ack_request parameter in the corresponding T_Data_Tag_Group.req service primitive automatically to the 'don't care' default value.

A_GroupPropValue_InfoReport.ind	(source_address, zone, object_type, object_instance, property_ID, data, *priority*, *hop_count_type*)
source_address:	Individual address of the sending device
zone:	Same value as in InfoReport.req
object_type:	Same value as in InfoReport.req
object_instance	Same value as in InfoReport.req
property_ID:	Same value as in InfoReport.req
data	Same value as in InfoReport.req
(hop_count_type):	dummy parameter
(priority):	dummy parameter

7.6.7 Restricted usage of Property_ID address range for LTE-HEE

Property_ID up to 154 are standardised Datapoint identifiers (globally defined behaviour) whereas Property_ID 155 … 255 are reserved for company specific (proprietary) Datapoints.

Company specific Properties are accessible only in client/server mode using individual addressing (point to point link) because the client knows the device type of server and the device type specific addressing and features of proprietary data.

Multicast distribution of company specific Datapoints using LTE-HEE services and Property_ID > 154 is not appropriate because the N receivers of the message do not know the device type of the sender and therefore the interpretation of the received private data is unknown.

Therefore for LTE-HEE messages only standardised Properties may be used. For private data additional mechanisms are defined as described below.

7.6.8 LTE-HEE private data

Runtime Interworking using LTE-HEE mechanisms with company specific data (in addition to the standard data!) must be possible. Companies want to have the possibility to implement additional company specific system features.

Requirements for private data:

- it is also absolutely necessary to keep the LTE zoning mechanisms
- within a standardised Functional Block (Interface Object Type) private Datapoints (Properties) must be possible
- it must be possible to implement proprietary Functional Blocks, i.e. proprietary Interface Object Types must be supported

Example

A boiler will send (A_GroupPropValue_InfoReport) standard process data with the binding link 'ProdSegment.Producer' according to the HWH Application Interworking Standard. In a multi-vendor system standard boiler functionality is provided.
In a single-vendor boiler cascade system, additional proprietary data will be exchanged to allow manufacturer specific extra functionality. The boiler will additionally send proprietary process data with the binding link 'ProdSegment.Producer'.

The following local service primitives are introduced for private data. They are mapped in the AL to existing A_Group_PropValue_....-PDUss (no additional APCI)

A_GroupPrivPropValue_Read.req	(zone, object_type, object_instance, priv_property_ID, company_code, ack_req, priority, hop_count_type)
A_GroupPrivPropValue_Read.ind	(source_address, zone, object_type, object_instance, priv_property_ID, company_code, ack_req, priority, hop_count_type) The application program shall check if the private data is available (additional company_code and priv_property_ID have to match).
A_GroupPrivPropValue_Response.req	(zone, object_type, object_instance, priv_property_ID, company_code, data, ack_req, priority, hop_count_type) Response is only generated by the application program if the Datapoint address information in ..._Read.ind was matching
A_GroupPrivPropValue_Response.ind	(source_address, zone, object_type, object_instance, priv_property_ID, company_code, data, *ack_req*, *priority*, *hop_count_type*) The application program shall check if the received private data is supported (additional company_code and priv_property_ID have to match).
A_GroupPrivPropValue_Write.req	(zone, object_type, object_instance, priv_property_ID, company_code, data, ack_req, priority, hop_count_type)
A_GroupPrivPropValue_Write.ind	(source_address, zone, object_type, object_instance, priv_property_ID, company_code, data, ack_req, priority, hop_count_type) The application program shall check if the received private data is supported (additional company_code and priv_property_ID have to match).
A_GroupPrivPropValue_InfoReport.req	**(**zone, object_type, object_instance, priv_property_ID, company_code, data, ack_req, priority, hop_count_type)
A_GroupPrivPropValue_InfoReport.ind	(source_address, zone, object_type, object_instance, priv_property_ID, company_code, data, ack_req, priority, hop_count_type) The application program shall check if the received private data is supported (additional company_code and priv_property_ID have to match).
company_code	globally defined manufacturer identification
object_type	either a standardised Interface Object Type or one of the reserved types for proprietary objects (see below)
priv_property_ID	private property identifier
Other parameters	same as in corresponding A_GroupPropValue_.... service

Encoding of LTE-HEE private data:

The field Property_ID is fixed to 255 in all A_GroupPropValue_.... messages containing private data. This is the criterion in the receiver to detect private data in the APDU and map the information to the corresponding A_GroupPrivPropValue_... service primitive

A) Private Properties in a standardised Interface Object Type

- Standard Interface Object Type ≤ 50 000
- Fixed Property_ID= 255 for private data
- First 2 data octets contain the globally defined Company Code to avoid message conflicts in multi-vendor systems.
- If an A_GroupPropValue_ReadPDU to a private property is received, this read request may be answered. If answered, the A_GroupPrivPropValue_Response-PDU shall be used.
- Following octets contain company specific data with a PrivProperty_ID + [data]

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15
Object Type		Object Instance=0	Property ID=255	Company Code		PrivProperty ID

Figure 37 - part of A_GroupPropValue_Read_PDU with private property

The Read-PDU for private properties contains 3 octets more information than a normal Read-PDU.

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15	Octet 16...N
Object Type		Object Instance	Property ID=255	Company Code		PrivProperty ID	Data

Figure 38 - part of A_GroupPropValue_Response_PDU with private property + data

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15	Octet 16...N
Object Type		Object Instance=0	Property ID=255	Company Code		PrivProperty ID	Data

Figure 39 - part of A_GroupPropValue_Write_PDU with private property + data

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15	Octet 16...N
Object Type		Object Instance	Property ID=255	Company Code		PrivProperty ID	Data

Figure 40 - part of A_GroupPropValue_InfoReport_PDU with private property + data

If an A_GroupPrivPropValue_Read-PDU to a standard Property is received, this read request may be answered. If answered, the A_GroupPropValue_Response–PDU shall be used.

B) Private Interface Object Types

- Reserved range of Interface Object Types 50 001 to 65 535
- Interface Object independent Properties (PIDs 1 to 50) shall always represent standardized data. Interpretation of Property data is not company-specific and therefore, these Properties shall not be encoded as private data within A_GroupPropValue-services.
- Fixed Property_ID= 255 for private data
- First 2 data octets contain the globally defined company code to avoid message conflicts in multi-vendor systems: Interpretation of the Interface Object Type depends on the company code
- If an A_GroupPropValue_Read–PDU to a private Property in a private Interface Object is received, this read request may be answered. If answered, the A_GroupPrivPropValue_Response-PDU shall be used.
- Following octets contain company specific data with a PrivProperty_ID + [data]

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15
Object Type (> 50 000)		Object Instance=0	Property ID=255	Manufacturer Code		PrivProperty ID

Figure 41 - part of A_GroupPropValue_Read_PDU with private Property

The Read-PDU for private properties contains 3 octets more information than a normal Read-PDU containing the address of a standardised Property.

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15	Octet 16...N
Object Type (> 50 000)		Object Instance	Property ID=255	Manufacturer Code		PrivProperty ID	Data

Figure 42 - part of A_GroupPropValue_Response_PDU with private property + data

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15	Octet 16...N
Object Type (> 50 000)		Object Instance=0	Property ID=255	Manufacturer Code		PrivProperty ID	Data

Figure 43 - part of A_GroupPropValue_Write_PDU with private property + data

Octet 9	Octet 10	Octet 11	Octet 12	Octet 13	Octet 14	Octet 15	Octet 16...N
Object Type (> 50 000)		Object Instance	Property ID=255	Manufacturer Code		PrivProperty ID	Data

Figure 44 - part of A_GroupPropValue_InfoReport_PDU with private property + data

8 Network management

For LTE devices there are no specific requirements concerning network management or deviations from standard modes.

8.1 RF Domain Address

The LTE RF BD device shall to this support the Management Server side requirements of the following Management Procedures (see [07]).

Via Programming Mode

- NM_DomainAndIndividualAddress_Write2, and
- NM_DomainAndIndividualAddress_Read

Via the KNX Serial Number

- NM_DomainAndIndividualAddress_Write3

8.2 Individual Address assignment

8.2.1 LTE TP1

Each LTE device has an unique individual address (mandatory) which is normally set by the installer once at installation time (normally using keys/display or a code wheel in simple devices). This address may also be read or changed by a tool (ETS) using standard NM mechanisms and services.

Subnetwork Address: medium dependent default value at manufacture.

- medium specific default value if only one subnetwork exists
- is assigned remotely by the corresponding router (standard Subnetwork Address update mechanism) or remotely by a tool
  ⇒ standard NM mechanisms and services as described in [08].

Device address: default device address (FFh) when unconfigured.

- Can be **set locally** (manual address management). This mechanism is recommended if the device has a local user interface.
  In case of manual address setting special care and considerations are necessary in mixed installations, especially if an LTE device is added to e.g. an S-Mode installation.
  ⇒ after address configuration, devices shall check the individual address before using it. Individual address check uses the same mechanisms like DAA, see [08].
- or **assigned remotely** from the network by a tool, address server etc.
  ⇒ standard NM mechanisms and services, see clause 8.4
- or **automatic device address assignment** using standard DAA mechanisms and services (self-acquisition; search for free address) as described in [08].
  This mechanism is <u>not</u> the preferred solution for LTE devices but it may be implemented on devices which do not have a local user interface.

8.2.2 LTE RF BD

8.2.2.1 General

Three methods for Individual Address assignment on RF are possible and at least one shall be supported.

1. Programming Mode
2. KNX Serial Number
3. Local Assignment

8.2.2.1.1 Programming Mode

The Individual Address (IANew) shall be assigned together with the RF Domain Address (DoANew) through the Network Management procedure NM_DomainAndIndividualAddress_Write2 (as already specified in 8.1) as follows.

```
NM_DomainAndIndividualAddress_Write(NmpDoANew = DoANew, NmpIACurrent, NmpIANew = IANew)
```

8.2.2.1.2 KNX Serial Number

Alternatively to the Programming Mode procedure described in the clause above, bidirectional LTE devices may be able to initiate the device individualisation procedure themselves. Note that the procedure starts by sending Device Descriptor Type 2 using the A_DeviceDescriptor_InfoReport service as specified in [05]. (This service is used on broadcast communication mode with AddressExtensionType = 0. Consequently, the frame contains the KNX Serial Number of the sender.) The frame with Device Descriptor Type 2 is received by a central management client that is in configuration mode.

At this point, the management client knows the KNX Serial Number of the device and can assign the Individual Address with the service A_IndividualAddress_SerialNumber_Write. Finally the Device Descriptor is read again in order to check the Individual Address of the device.

```
        /* The device spontaneously transmits the value of its DD2; this contains the CCs and the AppID. */
        /* The KNX RF frame is transmitted with AddrExtensionType 0 and shall thus contain */
        /* the KNX Serial Number of the device */
DM_DeviceDescriptor_InfoReport(DM_DDType = Type 2, DM_DD = DD2 of the device)
        /* The Management Client shall now assign the IA to the device using its KNX Serial Number. */
NM_IndividualAddress_SerialNumber_Write(NMIANew = IANew)
```

8.2.2.1.3 Local Assignment

Setting of the Individual Address via local user interface is also possible but in practice not very useful since the Domain Address is anyway assigned via a management client, which can also set the Individual Address.

If assigned locally the device shall check the Individual Address together with a valid Domain Address before using it (Individual Address check using standard mechanisms).

8.2.3 LTE RF Tx

Individual Address assignment is relevant for bidirectional devices only (LTE RF BD). Transmit only devices (LTE RF Tx) shall use the default Individual Address.

8.2.4 Unload IA for LTE Mode

The IA of LTE devices can be set via Programming Mode or via the KNX Serial Number procedure. By the fact that a LTE device has to support only one of the two methods, the Management Client has to check out this via a try and error method or via knowledge of the device.

If a device contains a KNX Serial Number then the KNX Serial Number procedure can be used.

Therefore the starting point is the KNX Serial Number procedure.

There are two possible procedures.

- The unloading of IA is done automatically; Programming Mode is activated by setting bit 0, ‚PROGMODE, of PID_PROGMODE (see [06]) in the Device Object; see procedure below.
- The unloading is done by setting the Programming Mode via user interaction on the device; the IA of the device is reset via NM_IndividualAddress_Write(IA = IA_{new}).
  (IA_{new}.SNA medium dependent default Subnetwork Address, see [06]).

Procedure: Unload IA automatically

```
IA_new.SNA = medium dependent default SNA
IA_new.DA = FFh
        ; The KNX Serial Number of the device (SN_Device) has to be known by the client from former actions
        ; to the device, if not then the KNX Serial Number is 000000000000h (invalid SN).
if SN_Device > 000000000000h
                ; Verify if the device with this KNX Serial Number exists.
                ; if no device with the given KNX Serial Number exists on the network there will be no answer.
        NM_IndividualAddress_SerialNumber_Read(SN = SN_Device)
        if positive response
                        ; Write IA_new using the devices's KNX Serial Number
                NM_IndividualAddress_SerialNumber_Write(SN = SN_Device, IA = IA_new)
                Exit procedure( )
        else
                Exit procedure (error = KNX Serial Number failed)
        end if
else
                ; From this stage the programming mode procedure starts for devices without KNX Serial Number.
                ; Check if devices in programming mode, if so, then the procedure will fail because the subsequent
                ; setting off the programming mode will cause two devices to be in programming mode
        NM_IndividualAddress_Read( )
        if responses
                Exit procedure (error = devices in programming mode)
        else
                        ; Set the device into programming mode using the IA of the device.
                A_PropertyValue_Write(DeviceObject,PID_ProgMode, value=1)     ; connectionless.
                        ; Set IA of the device to IA_new and then restart the device.
                A_IndividualAddress_Write (IA_new)
                A_Restart
        end if
end if
```

8.3 Device Identification

It shall be possible to identify each LTE-device on the bus, e.g. by a tool, using standard Network Management Procedures and Application Layer services.

LTE devices shall therefore

- support either one or both of the services A_DeviceDescriptor_Read and A_DeviceDescriptor_InfoReport, and
- support Device Descriptor type 0 (DD0) or Device Descriptor type 2 (DD2).

In [10] it is specified which services and which Device Descriptor type is mandatory for a given Profile.

NOTE 1 Currently, the following Profiles are specified for LTE:
- LTE TP1
- LTE RF BD
- LTE RF Tx

NOTE 2 The Profile “System 300” models the S-Mode Interface of the Profile LTE TP1 and only requires DD0.

If a Profile definition requires the support of Device Descriptor type 2, then all fields shall be supported in LTE Mode:

- Application Manufacturer
- Device Type
- Version
- Linking Mechanism
- 1st Channel-Code field : fixed to 1FF4h (=LTE)
- Other Channel code fields are void (or may contain meaningful channel codes if implemented in the device besides LTE)

Additional standard Interface Objects / Properties can be used to read out more descriptive information (e.g. ‘Logical Tag’ information).

8.4 Remote device configuration

Some devices do not have a local user-interface or are installed in the ceiling (e.g. HVAC-Terminal Unit devices). Therefore configuration of the individual address or LTE zones etc. locally on the device can in practice be a problem.

For devices without local user interface, configuration of the individual address and logical tags will be done by a “tool” (e.g. a service tool, handheld unit, Apartment MMI, Room MMI, ETS etc.). Since all logical tags are „normal“ Datapoints, they can be accessed in LTE devices as standardised Properties of Interface Objects.

- Remote configuration of logical tags by a “tool” is possible using standard property services after the device has a valid individual address.
- In addition the “tool” must know the type of device to be configured (all configuration Properties must be known e.g. from an off-line database in the “tool”)

Installation steps:

Unconfigured devices have the default individual address. The first step of device configuration is the remote assignment of the individual address by the “tool”.

At least one of the following standard mechanisms for remote individual address assignment shall be supported by LTE devices:

- **Assignment of the individual address in Programming Mode**:
  Device selection and indication of Programming Mode via local user interface (e.g. push-button and LED). The individual address is written to the device using the standard mechanism NM_IndividualAddr_Write

- **Assignment of the individual address via Serial Number**:

  How to get the serial number on the “tool”?

  - “EASY” mechanism
    The network can be scanned for all devices that have the default Individual Address using the standard mechanism NM_SerialNumberDefaultIA_Scan (see [07]).
  - LTE-HEE mechanism (using an existing LTE-HEE AL service, no new mechanism):
    The device shall send its serial number to the “tool” after a special event, e.g. when pressing a “programming” button on the device (see Programming Mode above). The device uses the LTE-HEE service A_GroupPropValue_InfoReport with LTE-HEE broadcast Group Address and containing the serial number which is a standard Property of the Device Object.
    This procedure is independent of Subnetworks / routers
  - The serial number of the device to be configured is entered on the “tool” e.g. by a barcode reader.

  The individual address is written by the “tool” to the device using NM_IndividualAddress_SerialNumber_Write mechanism (see [07]).

The 2nd step is the:

- **Configuration of LTE zones and parameters**:
  After setting of the individual address, each LTE zone or parameter of the device can be configured by the “tool” using standard A_PropertyWrite service.
  LTE zone setting and download may be hidden from the installer if the zoning information is obvious and already configured on the “tool”. E.g. use the Room MMI to configure the room temperature sensor (both share the zone Apartment.Room)

8.5 Group Address check

Group Address check is not supported by LTE-Mode Devices, see KNX Handbook, Volume 6 “Profiles”.

Group Address check for LTE-HEE Group Addresses is not needed.

8.6 Management of LTE-HEE Group Address tables

LTE-HEE Group Address tables are set-up by the local application program according to the configured logical tags. Any change of configuration leads to a new calculation of the HEE Group Address table which is set-up in the Data Link Layer by local management services. See clause 7.3

The LTE-HEE Group Address tables are accessible by e.g. a tool as a mandatory property of specific Group Address Table Interface Object (see [06]).

8.7 LTE linking procedures for RF transmit-only devices

8.7.1 General requirements

The partner device is a bidirectional device.

The ‘Unidir‘ attribute in the RF-Ctrl field is used in the linking procedure to identify transmit-only devices and adapt the LTE binding mechanisms accordingly

8.7.2 Normal conditions

Only one link can be established at the same time (sequential installation).

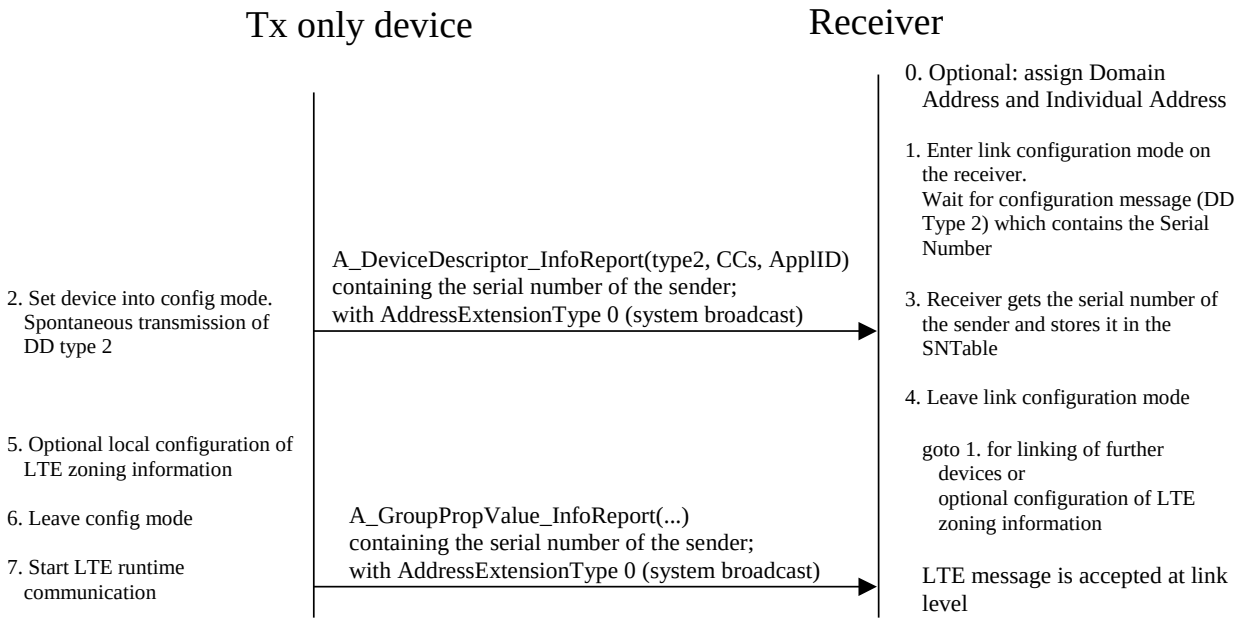


Figure 45 - Linking of transmit-only devices via KNX Serial Number

Steps

0. Bidirectional receiver: assignment of Domain Address and Individual Address

   Domain Address and Individual Address are on the receiver in principle not relevant for the linking of transmit-only devices. But it is recommended to set the Domain Address and Individual Address prior to the linking procedure.

1. Enter link configuration mode on the receiver.
   - ⇨ The mechanism is product specific, usually by pressing a button.
   - ⇨ The receiver attends a specific configuration message during a certain time-slot. If no configuration message is received the receiver will stop the configuration procedure after a product specific time-out.
2. Enter configuration mode on the sender.
   - ⇨ The mechanism is product specific, e.g. by pressing a button.
   - ⇨ The device sends the Device Descriptor Type 2 containing the Serial Number of the sender (system broadcast) using A_DeviceDescriptor_InfoReport service.
3. The message is received by the device in the link configuration state. The receiver stores the Serial Number in the Serial Number Table and uses it as a binding address for further 1:1 communication with the transmitter.

4. Afterwards, the configuration state in the receiver is closed automatically and the successful link is indicated to the user by e.g. an LED ⇒ 1:1 linking is complete.

   Further transmit-only devices can be linked on the receiver repeating steps 1 to 4. In addition, LTE zoning information may be configured on the receiver (optional for link with transmit-only device).

5. On the transmitter, optional LTE zoning information can be configured locally. If not supported, the sender will use default LTE zoning information that is together with the KNX Serial Number of the sender sufficient for unambiguous linking.

6. The transmitter leaves config. mode: product specific procedure either automatically or by local user interaction.

7. Transmitter starts LTE runtime communication (messages containing the KNX Serial Number of the sender ⇒ system broadcast). The Individual Address of the transmit-only device is not relevant and contains the default value.

Link modification

The same procedure as specified above shall be followed.

⇨ Modification of existing link

This shall be done by changing the LTE zoning information on the sender and receiver (if supported on the transmit-only device).

⇨ Deletion of an existing link

This can be done by

- physically removing the transmit-only device, or
- changing the LTE zoning information on the sender and establishing a link with a different receiver, or
- clearing the whole KNX Serial Number Table in the receiver and restarting linking of the installation, or
- removal of individual links from the KNX Serial Number Table; this is not foreseen in this specification; product specific solutions are allowed.

8.7.3 Error and exception handling

The receiver shall automatically leave the link configuration mode after a product specific time-out if no link message is received. Other messages than DD type 2 shall be ignored during linking state.

- Handling of full Serial Number Table and table entry overflow

  Please refer to the specification of the KNX Serial Number Table in [06].

- Handling of double entries in the KNX Serial Number Table

  If the same transmit-only device is linked twice with the same receiver only one KNX Serial Number Table entry shall be used. (Duplications shall be checked.)

- Life check of links

  Existing links shall not be removed automatically (cleared in the KNX Serial Number Table) by the receiver if no signals are received from the transmit-only device for a long time. In such a case the application program in the receiver may generate an error indication and provide product specific means to delete the link.

8.8 LTE linking procedures for RF bidirectional devices

8.8.1 Normal conditions

Linking of bidirectional devices shall be based on the Domain Address and LTE zoning information (same as on wired media).

The Domain Address can be considered as a higher level zoning information that extends conventional LTE zoning information.

Besides assignment of the Domain Address and configuration of LTE zoning information no further linking mechanisms are necessary.

Steps

1. One bidirectional device in the installation shall have the role of a Management Client (e.g. central unit) that assigns the Domain Address and Individual Address to the other bidirectional devices in the system.

   Both the Management Client and the devices to be linked shall be in configuration mode. To enter the configuration mode a product specific procedure shall be executed.

2. Assignment of the Domain Address to all bidirectional devices to be linked.
   See clause 8.1.

3. Assignment of the Individual Address to all bidirectional devices within the Domain.
   See clause 8.2.2.

4. Assignment of LTE zoning information on all devices to be linked. The mechanism is product specific. This can be
   - either decentralised on the local user interface of the device, or
   - centralised from a management client (central unit, service tool, ETS etc.). Setting of LTE zoning information shall be based on standard Properties that shall be accessed in point-to-point connectionless communication mode.

5. Configuration is complete and devices start LTE runtime communication.

Link modification

The same procedure as on wired media shall be followed.

- Change the Domain Address on all devices, e.g. if the central unit is replaced, see above. LTE zones shall not be changed.
- Modification of existing link within the Domain

  This shall be done by changing the LTE zoning information on the sender and receiver(s).

- Deletion of a link.

  This can be done by either
  - physically removing the device, or
  - changing LTE zoning information on the sender (e.g. zone ‘out of service’).

8.8.2 Error and exception handling

LTE runtime communication shall be inactive if the Domain Address is not configured (i.e. has the default value, see [06]).

There are no additional requirements in comparison with LTE mode on wired media.

8.9 Distribution of KNX Serial Number Table in the Domain

8.9.1 Normal conditions

If in a larger system with N transmit-only devices that have a 1:M relation with multiple receivers it is in principle possible to link all the N senders with M receivers separately. The link procedure as specified in clause 8.7 is executed many times and the KNX Serial Numbers of the 1 to N senders are memorised in M receivers.

Depending on the application and the number of transmit-only devices and the corresponding receiver(s) linking may be complex.

It may be more efficient and easier if all transmit-only devices are linked with one bidirectional device (e.g. a central unit) that will distribute the KNX Serial Number Table containing N entries within the Domain during configuration. With this mechanism all relevant KNX Serial Numbers of the transmit-only devices are then known within the Domain by all receivers.

This mechanism is optional and product or application specific.

- It reduces the configuration effort.
- The size of the KNX Serial Number Table in the receivers is usually larger because it will contain also irrelevant KNX Serial Numbers of devices that do not interwork with the receiver from an application point of view.
- LTE messages from the transmit-only devices of the same “application type” (having identical Functional Blocks) shall contain proper LTE zoning information in order to avoid ambiguous linking.

Distribution of the Serial Number Table

Since the KNX Serial Number Table is a Property of the Device Object it can be distributed within the Domain using the A_PropertyValue_Write-service and point-to-point connectionless communication.

The distribution of the table is selective at device level. If the size of the table exceeds the maximal length of the APDU, separate element(s) of the KNX Serial Number Table can be written individually. All KNX Serial Number Table elements in the receiver shall be written in order to have a consistent KNX Serial Number Table.

8.9.2 Error and exception handling

- KNX Serial Number Table size in the receiver is too small.

  The Management Client shall check the size of the KNX Serial Number Table in the receiver before writing it and generate an error indication of the table size is not sufficient.

- Identical FBs in multiple transmit-only devices using the same zoning information

  If multiple transmit-only devices contain identical types of Functional Blocks and the corresponding LTE messages contain only default LTE zoning information, the linking information may be ambiguous in the receiver (depending on the application). In such cases the receiver is unable to assign the messages properly without additional configuration effort on the sender or receiver. Handling of such zoning conflicts is product specific.

8.10 LTE Routers

LTE installations are not restricted to one Subnetwork. HVAC plants could consist of more than one Subnetwork in case of:

- multiple physical media: e.g. wireless individual room heating control sub-system based on Radio Frequency with TP1 backbone inside the apartment/floor/building
- need for multiple sub-networks due to bus-traffic in larger installations. Usually the router would be located between an apartment/floor and the building backbone. This router also provides better “privacy” of apartment internal data in residential buildings.
- better availability of the network: in residential applications a “manipulation” (like a TP1 short-circuit) on the network by the end-user will only affect one apartment

Handling of LTE-HEE messages:

LTE systems with multiple Subnetworks require routers with LTE-HEE message support.. For standard messages the standard router mechanisms apply.

Configuration of LTE routers:

- In “easy” LTE systems, LTE logical tags can also be used for router configuration. The LTE-HEE Group Address filter table is configured “automatically” according to the setting of LTE zoning information on the router. Especially the ‘geographical tag’ is useful as filter criterion.

  Application example: separating local traffic of an apartment
  For an LTE “apartment router” connecting one apartment with the building backbone (link to boiler and heat distribution) the Apartment-number would be configured to set-up the HEE Group Address filter table. LTE messages with the tag A.R.S on the Apartment subnetwork would not be transmitted to the backbone. On the other hand messages with the tag A.R.S on the backbone would be passed to the Apartment. Other group messages could be routed in both directions or could be filtered according to other configured tags.
- In complex LTE systems the LTE-HEE Group Address filter table is configured by a tool. The filter table is a property of a specific Interface Object in the router (LTE Address Filter Table Object).

Handling of Subnetwork Address setting and Subnetwork Address update in LTE-Routers:

Support of standard mechanisms as described in [08].

8.11 LTE linking procedures for RF BiBat synchronous devices

8.11.1 System configuration using zoning parameters

To set up the binding links, LTE devices shall provide zoning parameters (logical tags) that can be configured by the installer. LTE devices with the same zoning information within the same RF installation (Domain) can work together.

On TP media or asynchronous RF communication the LTE system configuration is usually fully decentralized and zoning parameters are set locally on each device.

In a LTE BiBat System LTE zoning parameters shall be set up centrally on the BiBat Master device, which shall always be present and which usually is a central unit.

- The zoning information and the corresponding LTE Group Addresses are used to assign BiBat group receive-block(s).
- The BiBat Master shall know the LTE zoning information of all involved BiBat devices. The BiBat Master will usually also be a central LTE system configurator where all the LTE zoning information can be set up.

8.11.2 Preconditions

- Configuration of LTE BiBat subsystem shall be done in a centralized way on the device containing the BiBat Master (central unit) because the BiBat Master shall know about the Domain Address, the Individual Addresses of the BiBat Slaves, the zoning information and the supported applications (Functional Blocks) and the time slots (receive-blocks) within the BiBat subsystem.
- LTE BiBat devices of the same RF installation shall share one common Domain Address. This Domain Address is normally the KNX Serial Number of the BiBat Master device in the installation that will distribute the Domain Address and Individual Address during "teach-in" procedure.
- LTE BiBat devices shall have an unambiguous Individual Address because of the following reasons.
  - Asynchronous upstream LTE messages of BiBat devices shall contain the source Individual Address of the sender.
  - Configuration of BiBat Slaves (LTE zoning information, BiBat receive-block table etc., application parameters) shall be based on property services.
- The BiBat Master shall have a list of the BiBat Slaves (device directory) in order to manage the assignment of individual receive-blocks (used for Individual Addressing only).
- Appropriate LTE zoning information shall be configured on the BiBat Master and BiBat Slaves in order to avoid zone conflicts during runtime Interworking.

  In addition LTE zoning information shall be used on the BiBat Master to assign the corresponding receive-blocks.

  Zoning information in the BiBat Slaves shall be configured remotely from the BiBat Master device. Zoning parameters shall be Properties of the corresponding Functional Blocks/Interface Objects. In order to write zoning information to the BiBat Slaves, the Domain Address and Individual Address of the BiBat Slave shall be assigned before.

- Application configuration

  The BiBat Master is not only a communication BiBat Master but shall also be the application BiBat Master in the BiBat subsystem. A LTE BiBat Master may of course support multiple applications.

  The BiBat Master device shall know a priori the possible partners (BiBat Slaves) and the corresponding Functional Blocks dedicated to its application(s).

  There is no need to enrol the functionality of the BiBat Slaves or check functional identifiers like e.g. "Channel Codes" before linking, because of the following grounds.

  - A LTE BiBat Slave with an expected functionality (supported Functional Block(s)) is either present (i.e. linked according to the zoning information) or not.
    During linking the zoning information of the expected corresponding Functional Block shall be written by the BiBat Master to the BiBat Slave using Property services. If the Interface Object / Property does not exist in the BiBat Slave, then the device does not match.
  - It is possible, that the BiBat Master provides optional LTE Properties to the BiBat Slaves that are not supported in by the implemented Functional Blocks of the BiBat receiver. The receiver shall ignore these Datapoints. This means unnecessary traffic / usage of receive blocks but there is no further problem.

- Assignment of receive blocks (individual, LTE group, broadcast) by the BiBat Master

  The BiBat Master shall also be application BiBat Master for the BiBat Slaves. Therefore the BiBat Master shall have knowledge about the information listed below.

  - application model, supported Functional Blocks, runtime-Interworking Datapoints and LTE zoning parameters in the BiBat Slaves; and
  - the number and type of downstream Datapoints using LTE broadcast or multicast addressing and standard Property services using Individual Addressing; and
  - the minimal number of messages per device and BiBat section and the related broadcasts, multicast and individual receive-blocks.

    NOTE The minimal number and type of receive-blocks per section is in principle predefined in the Application Model. However if a BiBat Slave may handle more receive-blocks due to higher battery capacity. This is indicated by the BiBat Slave during linking by the parameter PID_RECEIVE_BLOCK_NR, see PID_RECEIVE_BLOCK_NR in [06]

  Out of this information the BiBat Master shall be able to manage and assign the receive-blocks per BiBat Slave via the parameter PID_RECEIVE_BLOCK_TABLE.

  The receive-block table is assigned during linking using standard mechanisms, see PID_RECEIVE_BLOCK_TABLE in [06].

8.11.3 LTE linking procedures for BiBat devices

1. Set the BiBat Master into configuration mode.
2. Select the type of BiBat device to be linked on the BiBat Master according to the manufacturer specific user interface of the BiBat Master.

   EXAMPLE Subsequentially select application room heating control → room 1 → radiator 1.
3. Set the BiBat Slave (e.g. heating radiator controller) into configuration mode, e.g. by pressing a button.

   → The BiBat Slave shall spontaneously send its Device Descriptor Type 2 (asynchronous: procedure see 8.2.2.1.2).
4. Next shall follow the basic BiBat Slave Configuration Procedure steps 5..11 as specified in"Configuration of the BiBat Slave" in [02], this is, the BiBat Master assigns to the BiBat Slave
   - the Domain Address (see 8.1) and
   - an Individual Address (automatic selection in the BiBat Master), and
   - the values of the BiBat management properties.

5. Assignment of LTE zoning information by the BiBat Master.

   EXAMPLE Parameter *Room Nr = 1* to the Functional Block *HIRC*

   Setting of LTE zoning information shall be based on standardized Properties that are accessed in point-to-point communication mode (asynchronous communication).

   If the corresponding Interface Object/Property does not exist, this shall mean that the wrong device is attempted to be linked. This is an error.

6. The BiBat Master shall execute the Management Procedure DM_Restart_RCl() (see [07]) to restart the BiBat Slave.
7. The BiBat Slave shall send a help call.
8. The BiBat Master shall answer with a Help_Call_Response.
9. The configuration of one device is now complete and the device shall start LTE runtime communication (synchronous downstream and asynchronous upstream).
10. If the system configuration is not yet complete, next devices shall be configured repeating the procedure from step 2, else, the configuration procedure shall be ended.

8.11.4 Link modification

- Changing the Domain Address on all devices, e.g. if the Central Unit is replaced.

  ⇒ Continue like a new installation, see above.

- Modification of existing link within the Domain: e.g. replacement of a device.

  ⇒ Execute steps 2 to 8 for the devices that are concerned.

- Delete link: the action on the BiBat Master is product specific.
  - **Physically remove the device**

    ⇒ This may be detected automatically by the BiBat Master since the BiBat Slave does not send status information or does not respond to Property services in point-to-point connectionless or – connection-oriented communication mode.

    ⇒ The link can be deleted manually on the BiBat Master.

  - **Change LTE zoning information on the BiBat Master (e.g. zone 'out of service').**

8.11.5 Error and exception handling

Spontaneous LTE runtime upstream communication of BiBat Slaves shall be inactive

- if the Domain Address and Individual Address are not yet assigned (ex factory, void Domain Address 000000000000h), and
- during "teach-in" configuration procedure.

Synchronous downstream communication mode shall be inactive in the BiBat Slave

- if the receive-block table or the random pause table is not yet assigned (ex factory, void table value see PID_RECEIVE_BLOCK_TABLE in [06] and PID_RANDOM_PAUSE_TABLE in [06]), and
- during "teach-in" configuration procedure.
  In this case only asynchronous communication is activated.

Management of receive blocks by the BiBat Master

During linking of individual BiBat Slaves (one after another) the BiBat Master shall check the availability of remaining free receive blocks that can be assigned. Linking of further devices shall be disabled if the number of receive blocks is exceeded.

Linking of wrong device type (application mismatch)

This shall be checked by the BiBat Master during assignment of LTE zoning parameters in the corresponding Functional Block(s), see above.

9 LTE-HEE zone configuration (link management)

9.1 Remote access of logical tags

All LTE tags must be accessible as predefined/standardised properties of the corresponding Functional Block(s) / Interface Object(s) in individual addressing (client/server) mode (⇒ need for unique individual address in LTE devices)

A tool can then at least read out and visualise the configured tags. Usually also remote configuration of these tags is possible (write access to corresponding InterfaceObject.Property)

9.2 Void binding links

Complex LTE HVAC devices are often multipurpose / multifunctional devices. I.e. the firmware contains more Functional Blocks than needed in a certain application. The installer selects the required functionality by configuration (parameter setting) of the device and unused functions are disabled.

Example: a heating controller supports Domestic Hot Water control function which can be disabled by setting of a parameter. How the tag ‘DHWZone’ will be handled in this situation?

Generic mechanism

- For each type of logical tag a ‘void’ value is defined.
  I.e. the LTE zone is not active and LTE inputs/outputs in this zone are not activated.
- In order to have a generic mechanism, the standard STATUS/COMMAND data field and ‘OutOfService’ mechanism is used to encode ‘void’ zoning information
  ⇒ LTE zoning parameters have always the data type U_8Z_8 or $U_{16}Z_8$
  ⇒ see HVAC Datapoint Types in [09].
- Remark: usage of the tag value=0 as ‘void’ would not be appropriate because it may conflict with "wildcard" in structured zones

10 S-Mode Interface

10.1 Runtime-Interworking in mixed systems

Runtime-Interworking between devices using HEE mechanisms is a “cluster” and LTE-HEE frames are ignored by conventional S-Mode and E-Mode devices because of the different message formats (coexistence).

HEE uses the common Objects, but may have HEE specific Objects (own HEE Object list) for standard Data Point Types and Functional Blocks.

HEE provides an own run-time Interworking (incompatible to the Standard-System) as well as a Configuration Process which allow a multivendor approach within HEE, too.

In order to allow Interworking in mixed systems, a part of the LTE-HEE Datapoints shall be implemented additionally as standard S-mode group objects.

LTE products may bear the KNX Certification Mark if complying with both Standard-System-Interface and coexistence requirements as well as the HEE requirements.

Products complying with the HEE part only cannot bear the KNX Certification Mark.

Mandatory Objects (according to the application specification) have generally to be implemented both in HEE - and in Standard-System format. I.e. there are two access points (using Group Addressing) for the same Datapoint ⇒ see clause 6.4.

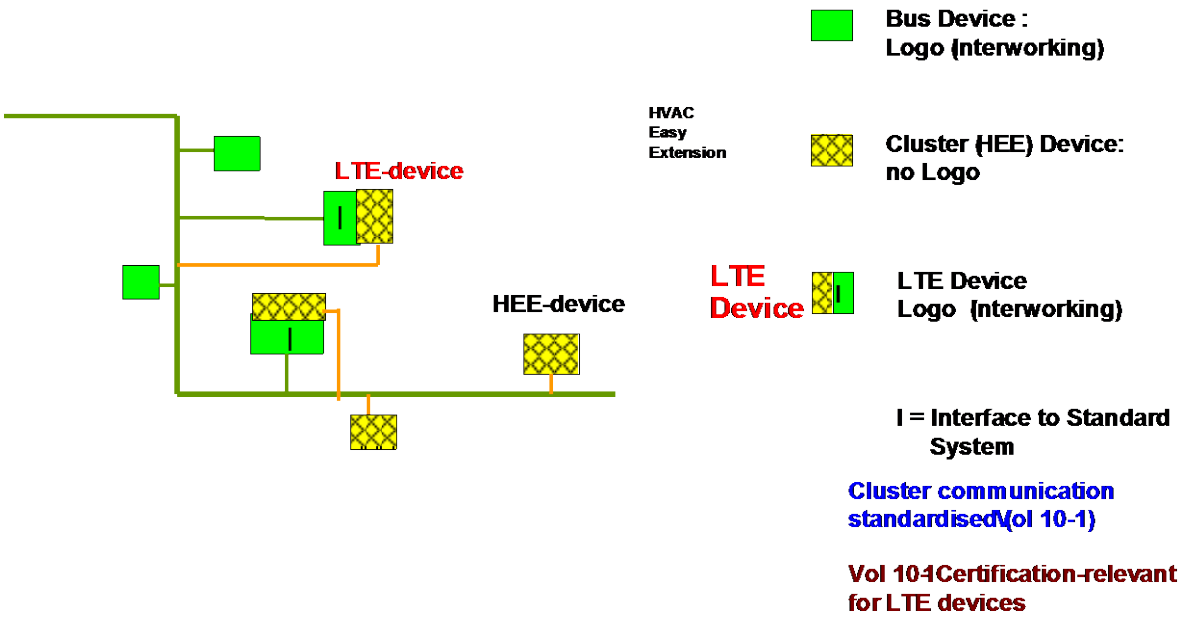


Figure 46 - LTE Standard System Interface

The S-Mode interface in LTE devices is based on:

- Standard mechanisms for configuration and runtime Interworking
- The Interworking with Standard-System is provided by Standard-System Datapoint Types
  ⇒ this may lead to data format conversions between HEE ↔ Standard Mode
- In mixed systems, HEE Datapoints which are also accessible in S-Mode can be activated by a tool (ETS). LTE devices are accessed by ETS as Standard-System devices and LTE devices support all necessary mechanisms and services for ETS tool access.
- It is the goal, that the future ETS also supports LTE/HEE devices (Realisation Concept tbd). .
- In unconfigured LTE devices, the S-Mode Interworking is usually "empty" (at manufacture). Exception may be for some special group objects of Functions of Common Interest (System Clock, Alarming).

Taking the S-Mode Interface as a base, other modes as e.g. Controller Mode can be implemented easily in addition.

10.2 Local mapping of S-Mode Datapoints to Properties of Interface Objects

The goal is to have a homogeneous software interface between Application Layer and the Application Program for HEE Datapoints as well as for S-Mode group objects.

Therefore standard group objects are mapped locally in the device (communication stack) to corresponding Properties of Interface Objects.

This is achieved by an extended Group Object Table that that replaces the standard Group Object Table. This table is part of the Extended Address Table Interface Object (see [06]). In this table every group communication object refers to a Property of an Interface Object of the device.

10.3 S-Mode interface of LTE RF devices

10.3.1 Bidirectional LTE RF devices (LTE RF BD)

Bidirectional LTE devices on RF shall provide the corresponding standard S-Mode Interface as on wired media.

The S-Mode Configuration of LTE RF BD devices is specified in [08].

10.3.2 Unidirectional LTE RF devices (LTE RF Tx)

It is accepted that LTE only runtime interworking is allowed in such devices without support of standard Group Objects because of the following reasons.

- Transmit-only devices cannot be accessed by ETS. Thus, remote activation of the S-Mode interface is not possible.
- Due to restrictions of duty-cycle and battery lifetime it is not practicable to send standard group messages in addition (e.g. fixed group messages according to E-Mode channels).
- It would not be practical to select either LTE-Mode or Standard Mode by e.g. local dip switch on the device.

However, the corresponding receiver shall map these objects to standard Group Objects as a proxy.

11 LTE device profiles

The LTE device profile is defined in Volume 6 "Profiles".

12 LTE Testing

Test Specification for LTE see KNX-Handbook, Volume 10 “Application Specific Standards”, Part 2 “System Conformity Testing”, Chapter 1 “Extended Frame Format (EFF) & Logical Tag Extended (LTE)”.

Annex A
(informative)
Application specific groups for HVAC-HWH

! simplified explanatory examples !

A.1 Heat production

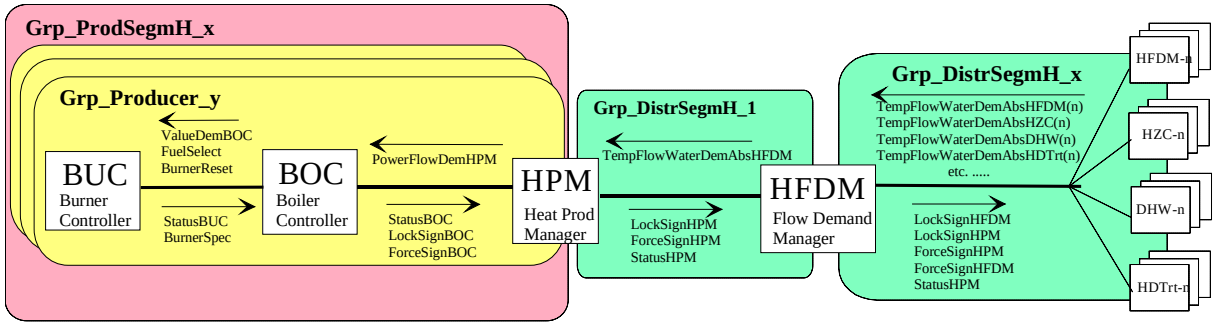


Figure 47 - HEE heat production

Data communication with the heat production in a distributed heating system has mainly 2 targets:

- – energy savings due to demand-dependent heat production
- – load management / load shedding in case of boiler overload or boiler overheat conditions

The figure shows a heat production segment containing one or more producers (boiler sequence) which are co-ordinated by the Heat Prod. Manager HPM. In a system with only one boiler the HPM functionality is reduced to a minimum.

Each producer contains a Burner Controller BUC and a Boiler Controller BOC which have a 1:1 relationship. They are often integrated into one device – otherwise BUC and BOC are linked by the group ProdSegmH_x.Producer_y.

The HPM receives the resulting overall heat flow demand in the heating system from the „first“ Flow Demand Manager HFDM in the primary Heat Distribution Segment. HPM and „first“ HFDM have always a 1:1 relationship and are usually located in the same device (and therefore dataflow between HPM and HFDM is normally purely device-internal).

The HPM controls the Producers according to the actual resulting overall heat flow demand (from HFDM) by sending the appropriate flow temperature demand to each Boiler Controller BOC which then controls the Burner Controller BUC accordingly.

For boiler overheat protection the BOC can send a forcing signal to the HPM.

Contrary, locking signals from BOC are used for boiler startup and overload protection.

Forcing and locking signals from each BOC are collected in the HPM and the resulting signals are passed by the “first” HFDM to the consumers and HFDM’s in the primary heat distribution segment.

Multiple independent Heat Production Segments are possible for completely independent heating sub-systems. This is a market need because houses with independent heating systems are today often linked by bus for remote management only. This feature is also interesting for DHW schemes with separate boilers for DHW generation.

A.2 Heat distribution and flow demand management

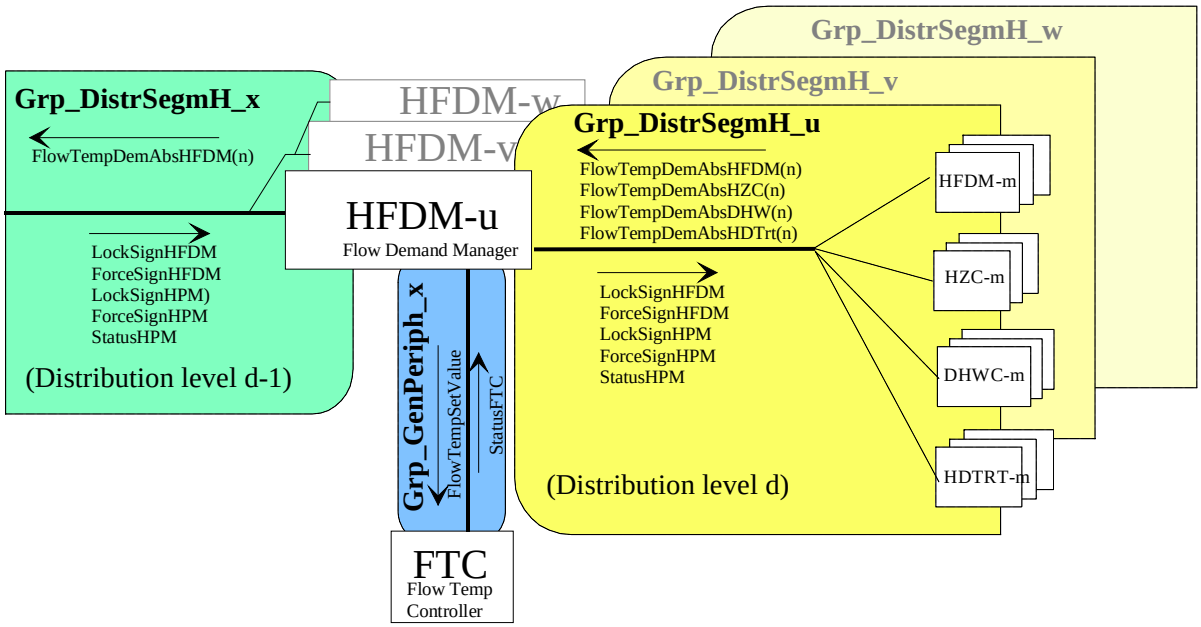


Figure 48 - HEE heat distribution

In more complex systems the consumers are not linked to the primary hot water Distribution Segment (directly connected to the boiler). Different levels of hot water distribution are possible (e.g. like high voltage - low voltage electrical distribution network). Each distribution level has its own hot water pipe.

The HFDM collects the flow temperature demand signals from all heat consumers (HZC, HDTRT, DHWC, HFDM) in the „right-hand“ Heat Distribution Segment (level d), calculates the resulting heat demand and sends it to the preceding „left-hand“ Heat Distribution Segment (level d-1).

The hot water flow temperature in the Heat Distribution Segment can be pre-controlled (by the FTC) according to the resulting heat demand of the consumers in the Distribution Segment. The Flow Temperature Controller and the Flow Demand Manager have a 1:1 relationship and are often located in the same device. Otherwise the 1:1 functional binding is established by setting a specific 1:1 unassigned peripheral link GenPeripheral_x

Distribution Segments may even be cascaded. In this case the resulting heat demand signals sent by the various HFDM’s (on level d) to the preceding „left-hand“ Distribution Segment (level d-1) are there collected again by an HFDM and the resulting heat demand is sent to the pre-preceding Distribution Segment (level d-2) etc...

In the other direction, the forcing and locking signals must be passed from each HFDM to the “right-hand” distribution segment

A.3 Individual room control

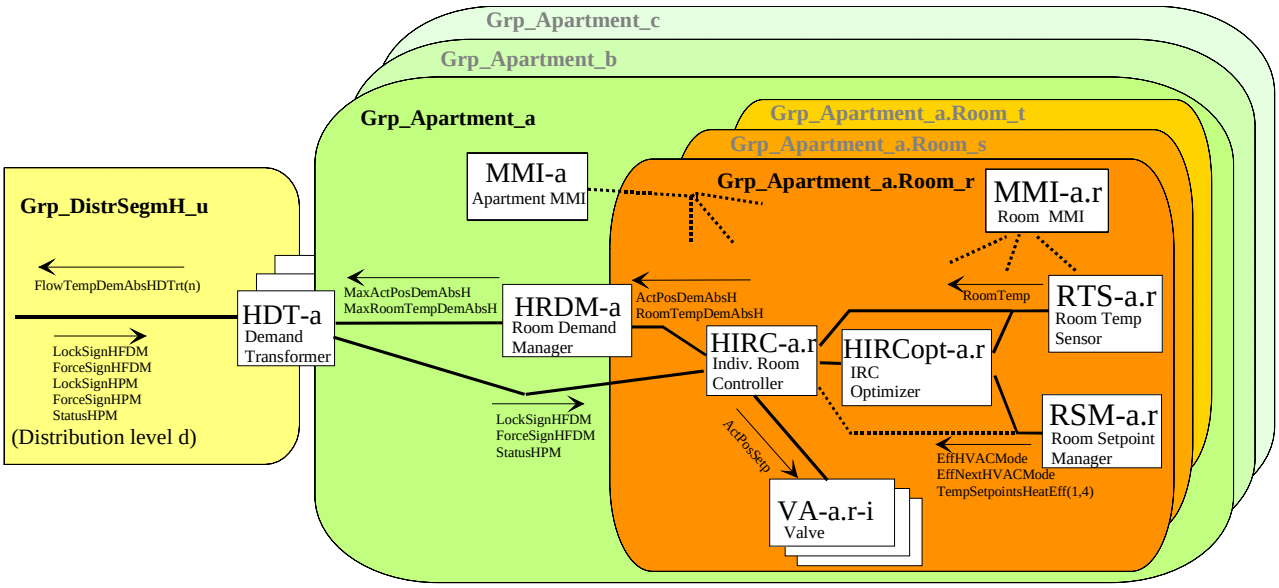


Figure 49 - HEE individual room control

This section describes individual room temperature control within a building-unit. A building unit is a cluster of rooms belonging together: in residential buildings e.g. an entire apartment or a single family home. In non residential applications a building-unit could be a floor etc. To simplify the description, the example of individual room control for apartments is described hereafter.

Room temperature control:

Each room in an apartment is controlled individually by an HIRC according to the actual room temperature setpoint provided by the Roomtemperature Setpoint Manager (RSM).

The actual room temperature setpoint in each room is calculated by the RSM and may depend on time schedule, heating operating mode, configured setpoints, presence detection, window status etc.

The HIRC calculates and controls the position of the valve(s) in the room. Within one room, usually all valves VA are controlled together by the HIRC. In mixed radiator / floor heating systems room-subzones are also possible.

Room demand:

The HIRC calculates the room demand to enable demand-dependant heat production. The heat demand may depend on the actual roomtemperature setpoint, actual room temperature value, valve position etc.

The Heating Room Demand Manager (HRDM) collects the room demands from all HIRC's in the apartment and calculates the resulting room demand which is transmitted to the Heat Demand Transformer (HDT).
The Heat Demand Transformer "translates" the resulting room demand to the corresponding hot water heat demand which is then transmitted to the HFDM in the Heat Distribution Segment.

The HFDM collects the heat demands from all apartments (HDT) and other consumers in the Heat Distribution Segment and calculates the resulting heat demand

User interface:
A room MMI can be used for remote control of the HIRC and may also contain the RSM and the room temperature sensor. A user interface in the apartment (apartment MMI) can be used for centralised remote control of the RSM's and HIRC's.

Annex B
(informative)
Example of an LTE-Mode device

B.1 Device Object

Interface Object Index = 0

Table 1 - Device Interface Object of an LTE-Mode device

Property Name	Memory Type [a]	Property Identifier (Property Datatype)	Optional/ Mandatory	Read/ Write	Value
Object Type	NV	1 = PID_OBJECT_TYPE (PDT_UNSIGNED_INT)	M	R	0 = device Object
Serial Number	NV	11 = PID_SERIAL_NUMBER (PDT_GENERIC_06)	O	R	00FDxxxxxxh
Manufacturer Identifier	NV	12 = PID_MANUFACTURER_ID (PDT_UNSIGNED_INT)	M	R	00FDh
Device Control	V	14 = PID_DEVICE_CONTROL (PDT_GENERIC_01)	O	RW	Init Value is 00h
Manufacturer Data	NV	19 = PID_MANUFACTURER_DATA (PDT_GENERIC_04)	O	R(W)	
Version	NV	25 = PID_VERSION (PDT_GENERIC_02)	O	R	DPT_Version
Routing Count	NV	51 = PID_ROUTING_COUNT (PDT_UNSIGNED_CHAR)	O	R(W)	06h
MaxRetryCount	NV	52 = PID_MAX_RETRY_COUNT (PDT_UNSIGNED_CHAR)	O	R	33h
Programming Mode	V	54 = PID_PROGMODE (PDT_UNSIGNED_CHAR)	O	RW	Bit 0 = Programming Mode
Product Identification	NV	55 = PID_PRODUCT_ID (PDT_GENERIC_10)	O	R	Manufacturer Specific device type
Max. APDU-Length	NV	56 = PID_MAX_APDULENGTH (PDT_UNSIGNED_INT)	O	R	55 octets
Subnetwork Address	NV	57 = PID_SUBNET_ADDR (PDT_UNSIGNED_CHAR)	O	R	
Device Address	NV	58 = PID_DEVICE_ADDR (PDT_UNSIGNED_CHAR)	O	R	

[a] V = Volatile
NV = Non-Volatile

B.2 Group Address Table for S-Mode (System 300)

Interface Object Index = 1

Table 2 - Address Table Interface Object

Property Name	Memory Type [a]	Property Identifier (Property Data Type)	Optional/ Mandatory	Read/ Write	Number of elements	Values
Object Type	NV	1 = PID_OBJECT_TYPE (PDT_UNSIGNED_INT)	M	R	1	1 = Address table Object
Load Control	NV	5 = PID_LOAD_STATE_CONTROL (PDT_CONTROL)	M	W R	1	for further Information see Load / State machines
Address Table Format 0		23 = PID_TABLE (PDT_UNSIGNED_INT[])	M	W R	3	1001h, 1002h, 1003h
[a] V = Volatile NV = Non-Volatile						

B.3 Extended Address Table 1 (System 300)

Interface Object Index = 2

Table 3 - Address Table Interface Object

Property Name	Memory Type [a]	Property Identifier	Type	Number of elements	Description
Object Type	NV	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	1 = Address table Object
Load Control	NV	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	1	for further Information see Load / State machines
Frame format	NV	51 = PID_EXT_FRAMEFORMAT	PDT_UNSIGNED_CHAR	1	04h Geographical Tag 1
Address Table Format-1	NV	52 = PID_ADDRTAB1	PDT_GENERIC_04[]	7	1234h FFFFh 1230h FFFFh 1200h FFF0h 1204h FF0Fh 1534h FFFFh 1530h FFFFh 1500h FFF0h
[a] V = Volatile NV = Non-Volatile					

B.4 Extended Address Table 2 (System 300)

Interface Object Index = 3

Table 4 - Address Table Interface Object

Property Name	Memory Type [a]	Property Identifier	Type	Number of elements	Description
Object Type	NV	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	1 = Address table Object
Load Control	NV	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	1	for further Information see Load / State machines
Frame format	NV	51 = PID_EXT_FRAMEFORMAT	PDT_UNSIGNED_CHAR	1	05h Geographical Tag 2
Address Table Format-1	NV	52 = PID_ADDRTAB1	PDT_GENERIC_04[]	3	2534h FFFFh 2530h FFFFh 2500h FFF0h

[a] V = Volatile
NV = Non-Volatile

B.5 Extended Address Table 3 (System 300)

Interface Object Index = 4

Table 5 - Address Table Interface Object

Property Name	Memory Type [a]	Property Identifier	Type	Number of elements	Description
Object Type	NV	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	1 = Address table Object
Load Control	NV	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	1	for further Information see Load / State machines
Frame format	NV	51 = PID_EXT_FRAMEFORMAT	PDT_UNSIGNED_CHAR	1	06h Application Specific Tag
Address Table Format-1	NV	52 = PID_ADDRTAB1	PDT_GENERIC_04[]	1	00A1h FFFFh

[a] V = Volatile
NV = Non-Volatile

B.6 Extended Address Table 4 (System 300)

Interface Object Index = 5

Table 6 – Group Address Table Interface Object

Property Name	Memory-type [a.]	Property Identifier	Type	Number of elements	Description
Object Type	NV	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	1 = Address table Object
Load Control	NV	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	1	for further Information see Load / State machines
Frame format	NV	51 = PID_EXT_FRAMEFORMAT	PDT_UNSIGNED_CHAR	1	07h Unassigned Tag
Address Table Format 1	NV	52 = PID_ADDRTAB1	PDT_GENERIC_04[]	1	0654h FFFFh

[a.] V = Volatile
NV = Non-Volatile

B.7 Association Table (System 300)

Interface Object Index = 6

Table 7 - Association table Interface Object

Property Name	Property Identifier	Type	Number of elements	Description
Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	Association table Object 2
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	1	for further Information see Load / State- machines
Association Table	23 = PID_TABLE	PDT_GENERIC_02	3	01h 01h 02h 02h 03h 03h

B.8 Application Program Object (System 300)

Interface Object Index = 7

Table 8 - Application Program Interface Object

Property Name	Property Identifier	Type	Number of elements	Description
Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	Application Object 3
Load Control	5 = PID_LOAD_STATE_CONTROL	PDT_CONTROL	1	Read-only Value is LOADED
Run Control	6 = PID_RUN_STATE_CONTROL	PDT_CONTROL	1	for further Information see Run / State machines
Application Version	13 = PID_APPLICATION_VERSION	PDT_GENERIC_05	1	00h FDh 01h 01h 01h

B.9 Group Object Table Object (System 300)

Interface Object Index = 8

Table 9 - Group Object Table Interface Object

Property Name	Property Identifier	Type	Number of elements	Description
Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	Group Object Table
Object Table	51=PID_GRPOBJTABLE	PDT_GENERIC_06[3]	3	Group Object Table
Ext. Object Table Ref	52=PID_EXT_GRPOBJTABLEREF	PDT_GENERIC_08[3]	3	Extended Group Object Table Reference

Group Object Table

2 octets	2 octets	2 octets
Configuration	Data Type (High part of DPT)	Data Type (Low part of DPT)
00FFh	5 = 8 Bit Unsigned	1
00FFh	7 = 16 Bit Unsigned	1
00FFh	…	..

Extended Group Object Table Reference

2 octets	1 octet	1 octet	2 octet		1 octet	1 octet
Object Type	Object Instance	Property Identifier	Reserved (4 bit)	Start Index (12 bit)	Bit Offset	Conversion
2222	1	55	0	1	0	0
2222	1	56	0	1	0	0
2222	..					

B.10 First application specific object (System 300)

Interface Object Index = 9

Table 10 - Application Interface Object 1

Property Name	Property Identifier	Type	Number of elements	Description
Object Type	1 = PID_OBJECT_TYPE	PDT_UNSIGNED_INT	1	Value = 2222
…	55	PDT_UNSIGNED_CHAR	1	
…	56	PDT_UNSIGNED_INT	1	

