



Whitepaper

KNX IoT System and Application Configuration

KNX IoT, an introduction to KNX IoT applications and network concepts

Version 0.1.2, May 1, 2025

Copyright: KNX Association & Siemens Schweiz AG, 2025

Authors: Oskar Camenzind

Status: Published

Classification: Public

Contents

1	Introduction	3
1.1	Purpose of this document	3
1.2	Intended Audience	3
1.3	Summary	3
2	Motivation	4
2.1	KNX System Extension with KNX IoT	4
2.2	Runtime Interworking	5
2.3	Converged IP Stack	6
3	KNX IoT Point API	7
3.1	Communication Stack	7
3.2	Point & Object Layer	8
3.3	Transport Binding	11
4	Network Topology	13
4.1	KNX Network Topology	13
4.2	Vertical Integration	14
4.3	Horizontal Interworking	16
5	Application Engineering	19
5.1	Application Functions	19
5.2	Device Interworking Patterns	21
6	Device Configuration	22
6.1	Function Point Tables	22
6.2	Access Control List	27
7	Conclusion	28
8	References	28

1 Introduction

1.1 Purpose of this document

The purpose of this whitepaper is to outline the challenges, solutions, and benefits associated with KNX IoT, with a specific focus on leveraging existing IP technologies, standards and explaining how the KNX IoT Point API supports various interworking models. By bridging KNX Classic and KNX IoT, the document demonstrates how scalable, secure, and interoperable solutions can effectively meet the evolving demands of HBES/BACS (Home and Building Electronic Systems / Building Automation and Control Systems). KNX IoT places a strong emphasis on the importance of semantic data and seamless integration, not only within the KNX ecosystem but also with non-KNX ecosystems.

The first part introduces the motivation behind this document, the KNX IoT communication stack, and the structure of KNX IoT networks. It offers an overview of the KNX IoT system and explains how to configure group objects using a tool.

The second part delves into a detailed explanation of communication patterns and their configuration within the Function Point Table. As the Function Point Table is a central concept in KNX IoT interworking, this section is particularly valuable for readers seeking a deeper understanding of specific device configurations.

1.2 Intended Audience

The purpose of this whitepaper is to provide device manufacturers, developers, product managers, system integrators, installers, and business leaders with a comprehensive overview of KNX IoT. This document is designed to help stakeholders see the big picture, grasp core concepts, gain a deeper understanding of specific aspects of KNX IoT, and prepare for upcoming changes within the evolving KNX ecosystem.

This document is intended for a wide range of stakeholders and aims to provide a comprehensive resource that supports knowledge sharing and collaboration within the building automation industry and especially the KNX ecosystem. Therefore, the target group of this document is manifold:

Device manufacturers will gain a clear understanding of the communication stack and how the Point API leverages existing KNX concepts.

Device developers will acquire detailed knowledge of KNX IoT, learn various message exchange patterns, and understand how to configure Function Point Tables.

System integrators and installers will receive an overview of the general setup, including the key differences between KNX IoT and KNX Classic, and how these two network segments can be seamlessly integrated within a single installation.

Product managers will benefit from insights into the potential functionalities that can be implemented on the devices they manage, enabling them to make informed decisions. They will also gain technical knowledge to create audience-specific documents or whitepapers.

Business leaders will gain a high-level understanding of the latest advancements and extensions related to KNX, helping them stay ahead in the competitive building automation market.

1.3 Summary

IP-based technologies like SPE (e.g., Single Pair Ethernet 10BASET1S) in control cabinets and Thread within rooms are now available, challenging KNX Classic in terms of interoperability, bandwidth and cost. KNX IoT extends the traditional KNX system by utilizing modern IP-based Internet technologies, such as IETF and IEEE

standards, and integrating them into the existing tools and concepts of the KNX ecosystem. This enables seamless interoperability between KNX Classic and KNX IoT.

The system architecture not only supports horizontal interworking between KNX IoT and KNX Classic, configured via ETS (KNX Engineering Tool Software, common tool for all product providers), but also enables vertical integration with third-party automation, control systems or services. Simple vertical integration allows either individual devices or entire systems to be easily integrated without requiring additional configuration in ETS. To simplify this integration, standard technologies are utilized, with open-source libraries readily available.

KNX IoT is specifically designed to also operate in constrained devices and networks and uses established industry standards such as CoAP, IPv6, OSCORE, and TLS to ensure secure and reliable communication across both wired and wireless infrastructures. This allows third-party clients to effortlessly access the REST data model of a KNX IoT Point API device, supported by ETS's semantic project export. The KNX IoT Point API uses REST concepts to facilitate horizontal interworking through event-driven group communication and for vertical integration through client/server models.

In the KNX ecosystem, Group Objects (GO) and Group Addresses are key concepts that enable horizontal interworking between two or more devices. Each KNX device typically includes at least one logical input and/or output, known as a Group Object. These Group Objects have configurable communication behavior, access control settings, and support various message exchange types to send messages from an output Group Object to an input Group Object.

The KNX IoT Point API supports three types of message exchanges: Multicast Message Exchange, Direct Unicast Message Exchange, and Endpoint Subscription Message Exchange, all of which are configured through the Function Point Table using ETS. Multicast Message Exchange enables a single device to send messages to multiple devices simultaneously, making it highly efficient for group-wide actions, such as turning on all lights in a room. Direct Unicast Message Exchange facilitates one-to-one communication, ensuring the reliable delivery of commands to specific devices, like adjusting the setpoint of a single actuator. Endpoint Subscription Message Exchange enables a receiving device to subscribe to changes from a sender through the CoAP Observe mechanism and receiving sender's real-time update.

By configuring the Function Point Table with ETS, system integrators can define how devices communicate using these patterns, ensuring seamless and efficient interaction within IPv6-based building automation systems that span multiple network segments and cybersecurity zones. From ETS version 6.3 on, Multicast Message Exchange is supported, while the other message exchange types will be gradually added in next ETS versions. To simplify the work of system integrators, the goal is to ensure that the configuration of the various types of message exchanges combined with the necessary permissions is supported by ETS, with the corresponding mechanisms automatically applied based on the concepts of Smart Linking. Smart Linking is a mechanism in ETS that automatically links input and output Datapoints of devices.

2 Motivation

2.1 KNX System Extension with KNX IoT

KNX has a strong footprint in the B2B market (typically midsize/large residential and commercial buildings) with more than 500 training centers worldwide for system integrators and installers. This market requires an efficient application commissioning, re-configuration, or device replacement of many devices in a decentralized system.

An important aspect of KNX IoT is the utilization of existing tools, applications, and concepts, allowing for a seamless expansion of a KNX system with KNX IoT devices that employ common Internet technologies based on IETF and IEEE standards. The KNX Association (KNXA) provides tools and specifications that enable integration of a KNX system or individual devices into other systems through standardized APIs.

2.1.1 IT/OT Convergence

The reuse of IT technology, such as IP-based communication, ensures compatibility between systems and shortens innovation cycles. Reusing existing IP-based technologies, hardware, and established standards, such as IEEE and IETF, provides cost advantages for manufacturers and system integrators.

Increased data flow and communication requires a network with higher bandwidth. For instance, firmware updates can be done significantly faster on IP-based networks compared to KNX-specific cabling (TP1), which can sometimes take up to an hour for a larger application program.

With the advent of new IP technologies like Single Pair Ethernet (SPE) [IEEE 802.3cg], for wired devices using the 10BASET1S and/or 10BASET1L technology within cabinets and Thread, commonly employed for wireless devices in rooms, along with increasing regulatory focus on cybersecurity, KNXA has developed KNX IoT to address the growing integration of IT and OT (Operational Technology = Building Automation) systems. KNX IoT enables the reuse of existing IT infrastructure and minimizes the need for additional field bus networks and cabling.

2.1.2 Data-Driven Services

Regulations that target reduced energy consumption and carbon emissions (e.g., EPBD) are driving the adoption of energy- and resource-efficient building automation solutions in both new constructions and retrofits. This shift leads to a growing demand for real-time data access and advanced analytics, either deployed on edge devices or in the Cloud. To address these needs, it is essential to enable efficient data utilization and optimize operations across interconnected ecosystems. Vertical integration with direct access to device data is key to unlocking these capabilities and ensuring seamless functionality. Additionally, it facilitates remote maintenance, which not only helps optimize energy consumption but also reduces maintenance and commissioning efforts.

2.2 Runtime Interworking

KNX IoT is built on three building blocks that work together to enable seamless integration and runtime interworking. The first is **Vertical Integration**, which allows third-parties to integrate devices through the KNX IoT Point API without the need for ETS, streamlining the connection of KNX IoT devices with third-party systems or Cloud services. The second is **Horizontal Interworking** of KNX Classic and KNX IoT devices, enabling direct IPv6 communication between sensors and actuators for building applications, along with efficient group messaging for reliable device-to-device communication. ETS is used to configure group communication for KNX device interworking. Finally, the **ETS Semantic Project Export** provides detailed project data, such as device locations, functions, and interdependencies, in a JSON-LD format, making configuration and integration more efficient. The diagram below illustrates how these building blocks interconnect within a KNX system.

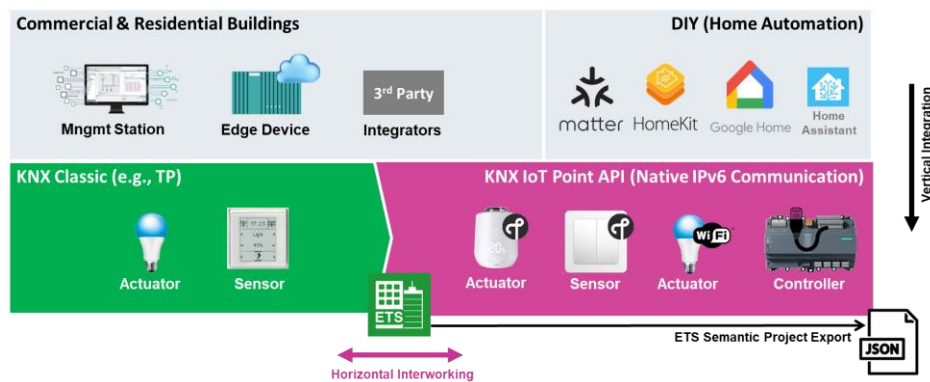


Figure 2-1: The image illustrates a KNX Classic system connected to a KNX IoT system. Horizontal interworking between KNX devices is configured using the ETS. Additionally, the KNX system can be vertically integrated with non-KNX systems or devices.

2.3 Converged IP Stack

Communication between two humans necessitates a common structure and shared understanding. From whales to humans, most mammals employ communication that is based on a structured framework. Each language adheres to distinct rules and structures. The same is true for Internet communication or OT system communication.

A communication protocol stack, often simply referred to as a protocol stack, is a comprehensive set of communication protocols arranged in layers to facilitate communication between devices or systems in a network. Each layer of the stack performs specific functions, and the combination of these layers form a protocol stack that governs the exchange of information. The OSI model is composed of seven functional layers, each with a specific role. Some layers are mandatory, while others are optional, depending on the use case. In the OSI framework, data flows up or down the stack as it moves through the network. The simplified model presented here focuses on just three key layers from the OSI model: the Application, Network, and Data Link layers.

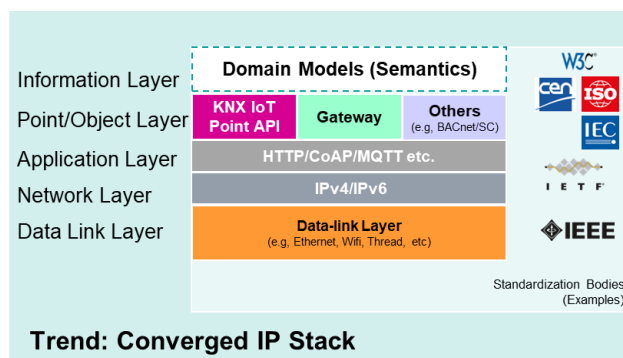


Figure 2-2: The Converged IP Stack consists of the following key layers: Information Layer, Point/Object Layer, Application Layer, Network Layer and Data Link Layer. The picture also shows which standardization organizations define the corresponding layer.

While the OSI model is used to send data across the network, the information model is used to semantically describe the data and define the relationships between the pieces of information. The Converged IP stack provides a structured framework that ensures data is not only transmitted but also understood in a meaningful context, enabling interoperability and coherent communication between devices and systems.

To implement the Converged IP Stack, KNXA has specified the KNX IoT Point API, which defines how the OSI layers are utilized. By leveraging IP, it becomes possible to reuse existing wireless or wired data link layers. At the Information Layer, KNX utilizes the Home and Building Electronic Systems Information Model [EN50090-6-2], which serves as the foundation for semantic data exchange (e.g., project export/import) and smart linking functionalities.

3 KNX IoT Point API

The following sections describe the KNX IoT Point API Communication Stack, and the technologies used for the respective communication layers. Additionally, the core concepts of the REST data model and [Point API Schema] are explained, along with how Datapoints relate to group objects used for KNX group communication.

3.1 Communication Stack

The KNX IoT Point API communication stack is built on existing standards, ensuring system openness and compatible communication across different functions. The device discovery uses the known protocols DNS-SD and mDNS [RFC 6762]. For cybersecurity, it follows recognized standards such as OSCORE [RFC 8613] and SPAKE2+ [RFC 9383]. There's an option to use TLS [RFC 4492] Transport Layer Security for device configuration or vertical integration. KNX IoT extends KNX Classic group communication with the KNX IoT Point API devices and uses CoAP/UDP. Read/Write/Publish/Subscribe functions enable a smoothly integration into non-KNX systems, following standard CoAP [RFC 7252], with support for content formats like CBOR [RFC 8949] and JSON [RFC 8259] if needed. This commitment to existing standards ensures a reliable and widely accepted foundation for communication in the system.

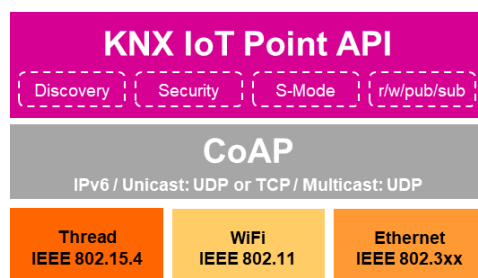


Figure 3-1: The protocol stack illustrates various data link layers, the network layer utilizing UDP and IPv6, and CoAP as the application layer. The KNX IoT Point API (Point / Object Layer) has different building blocks for device discovery, secure KNX group communication (S-Mode=System Mode), and read/write/publish/subscribe interworking.

The KNX data model is designed to facilitate communication between devices in a building automation system. It defines how data is represented, transmitted, and interpreted within the KNX network. Volume 7 of the KNX Specification defines a standardization application model and describes these, grouped per application domain: lighting, shutters and blinds, HVAC, and access control. The application model is described with functional blocks with standard properties and detailed behaviour to the level necessary to guarantee interworking. This means that devices from different manufacturers can communicate seamlessly within a KNX system, promoting interoperability. KNX also describes the interworking with other systems such as M-Bus or Dali.

3.2 Point & Object Layer

The KNX IoT Point API device supports two runtime interworking patterns: client/server communication with Datapoint read, write, publish, and subscribe (r/w/pub/sub) and group communication (System Mode) as known from classic KNX, which follows an event-driven sender/receiver communication pattern. The client/server communication pattern is used for device configuration or for vertical integration, for example, with third-party devices, while group communication is designed for horizontal interworking and requires configuration with ETS.

3.2.1 REST Data Model

The KNX IoT Point API data model is based on REST concepts where a Point represent an interface to a device functionality (see also [Point API Schema]). Points can be distinguished into system Datapoints to configure a device and application Datapoints. The application Datapoints are used for various application functions within a building automation system, such as switching lights, controlling heating, or monitoring sensors.

A KNX IoT Point API device comprises a list of functional blocks, each representing a software functionality with input ("if.i"), output ("if.o"), parameter ("if.p"), and diagnostic ("if.d") Datapoints. KNX IoT Datapoints contain additional metadata information. Input and output Datapoints can be configured as Group Objects, and a tool (ETS) can associate a Function Point to facilitate data exchange through KNX group communication with another device.

The following example illustrates a KNX functional block for a room temperature sensor (RTS) as defined in Volume 7 of the KNX Specification. The table presents two types of Datapoints. The first is an output Datapoint that also serves as a diagnostic Datapoint, allowing both the current status and its metadata to be accessed. The second is a parameter Datapoint, which is specifically used to configure a temperature correction value.

Functional Block ID	Property ID	Metadata	Value
321 (RTS)	51	Value	20.3
		id	knx://sn.<serialnumber>/p/rts/temproom
		href	/p/rts/temproom
		rt	dpa.321.51
		if	["if.d", "if.o"]
		desc	Room Temperature
		dpt	dpt.valueTemp
		unit	DEG_C
		min	-50.0
		max	50.0
		mrt	2
		cov	0.5
		hbt	900
	111	value	0
		id	knx://sn.<serialnumber>/p/rts/tempcorrvalue
		href	/p/rts/tempcorrvalue
		rt	dpa.321.111
		if	"if.p"

Functional Block ID	Property ID	Metadata	Value
		desc	Temp. Correction
		dpt	dpt.valueTemp
		unit	DEG_C
		min	-5.0
		max	5.0

Volume 7 of the KNX Specification defines, among other categories, the following KNX functional block types: System Clock, Common Sensors, Actuators and Controller Functions, Scheduler and Calendars, General Purpose I/O, Lighting, Shutter, Hot Water, and more. These functional blocks, along with their properties and corresponding Datapoint Annotations (DPA), serve as the foundation for runtime interoperability, Smart Linking, and other functionalities.

The following CoAP request demonstrates how to read (GET) the room temperature value from a device using its IPv6 address. In this example, the value is accessed from the resource `/p/rts/temproom`. While clients typically read only the current value, this example shows the possibility of retrieving all associated metadata as well. A comprehensive description of the device resources and their corresponding metadata can be found on the KNX IoT Point API Scheme Git page [Point API Schema].

REQUEST:

GET coap://{ipv6-unicast}/p/rts/temproom?m

Content-Type: application/json

RESPONSE:

Content-Format: application/json

```
{
  "value": 20.3,
  "id": "knx://sn.<serialnumber>/p/rts/temproom",
  "rt": [":dpa.321.51"],
  "if": [":if.d", ":if.o"],
  "dpt": ":dpt.valueTemp",
  "unit": "DEG_C",
  "min": -50.0,
  "max": 50.0,
  "mrt": 2,
  "cov": 0.5,
  "hbt": 900
}
```

3.2.2 Group Communication

KNX group communication is event-driven, meaning that devices send messages only when there is a change in status or when an event occurs. This helps to minimize unnecessary communication and reduce power consumption. Messages exchanged between devices include information such as the source device, a group address, the communication type (read, write, or response), and the associated Datapoint value. Devices have input or output Group Objects (GOs), each associated with a device internal Datapoint and a specific Datapoint Type (DPT). With ETS, an output GO from one device can be linked to input GOs on other devices. This connection is referred to as a Function Point.

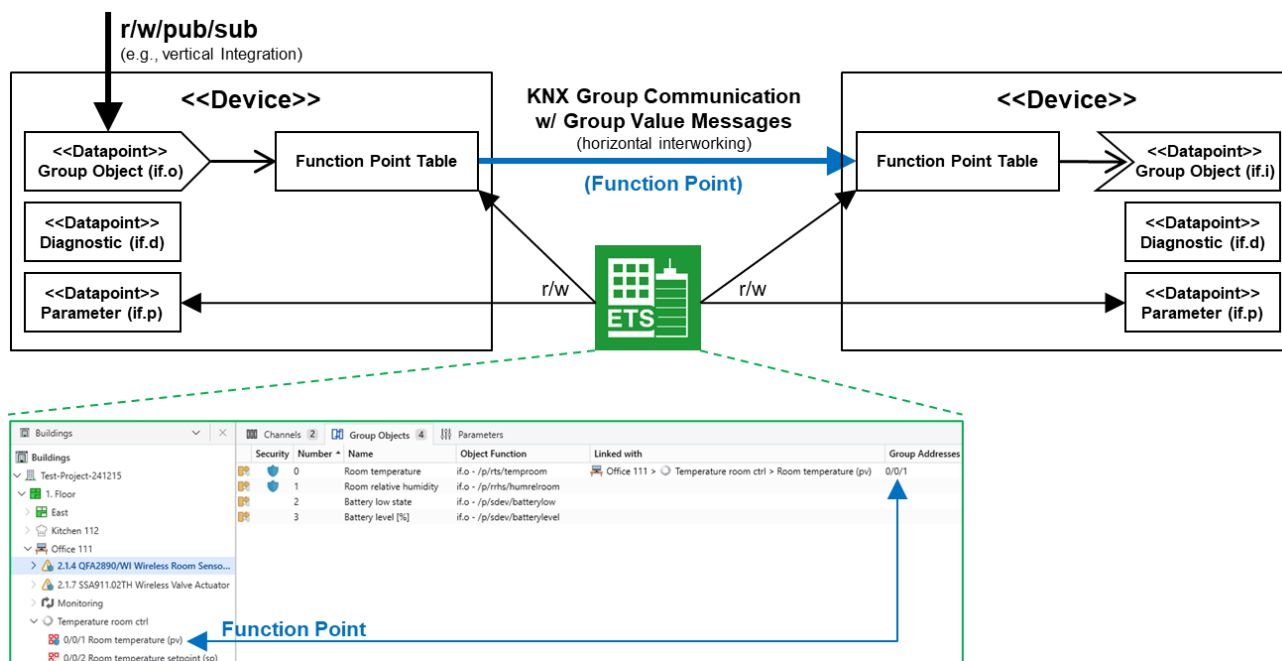


Figure 3-2: The figure illustrates two devices, each containing Function Point Tables with links to a device internal Datapoint. The ETS location view shows the sending Group Object (if.o) and its relation to the Function Point with the corresponding Group Address (0/0/1). Additionally, direct interaction with Datapoints is also possible, as indicated by the r/w/pub/sub (read, write, publish, and subscribe) arrow in the diagram. ETS uses the same mechanism to read and write (r/w) device parameters and Function Point Table configurations.

The Function Point Tables link Group Object (input or output Datapoints) with a Function Point's group address. When an event occurs, the Datapoint value is transmitted from the output Group Object of one device to the input Group Object of the other, effectively updating the associated internal Datapoints. The Group Value message transmitted between devices includes the Datapoint value, Group Address, and the communication type (read, write, or response). Furthermore, the multicast and unicast destination or recipient IPv6 address is configured with the Function Point Tables.

3.2.2.1 Group Communication Model

In the KNX ecosystem, Group Objects (GO) and Group Addresses are important concepts for the interworking of two or more devices. Each KNX device normally has at least one logical input and/or output, which is referred to as a Group Object. These Group Objects have configurable communication behavior, and each GO represents a Datapoint of a device.

When a Group Address is assigned to a Group Object, the Group Object becomes a member of a particular Function Point that contains a Group Address. A Function Point represents a link between output and input Group Objects. It is important to note that a Group Address represents the instantiation of a Function Point and is unique within the entire installation.

The flexibility of KNX devices allows for the assignment of input Group Objects to one or more Group Addresses. On the other hand, output Group Objects can also be assigned to multiple Group Addresses to transmit the value of the same datapoint of a device.

The following example shows two devices with input and output Group Objects. In this example, an installer has assigned three Channels to an Application Function. Their corresponding Group Object, for example, have

the same Datapoint Type (DPT) and Datapoint Annotations (DPA), therefore, the linking between output (if.o) and input (if.i) becomes possible. A Channel is a manufacturer-defined logical grouping of Group Objects in ETS (tool), typically used to distinguish a functionality that is used, for example, for Smart Linking. However, the concept of Channels does not exist on the device in the same way.

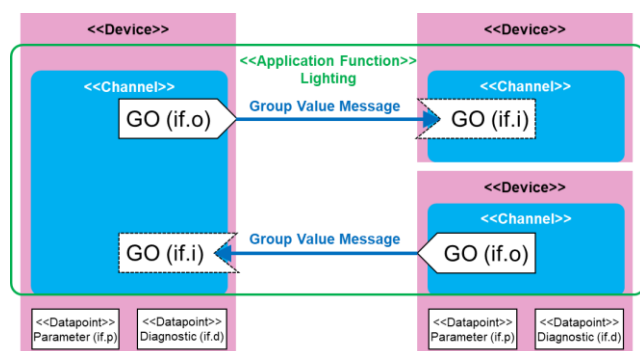
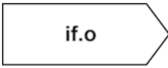
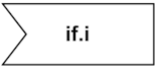
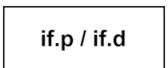


Figure 3-3: Group communication model between Group Objects with Group Value messages.

To ensure the desired behavior of the KNX device, configuration parameters may be present. These parameters determine specific functionalities and behaviors either for the entire device or for a specific device functionality such as a Channel. Each parameter is assigned a unique resource (href), which is referenced within the device.

Parameter and diagnostic properties are collectively referred to as Datapoints throughout this document. It is important to note that in KNX IoT (see also [EN50090-6-4]), properties are also referred to as Points and utilize the same communication mechanisms as runtime interworking Points (CoAP GET, PUT and POST). In contrast, in KNX Classic, diagnostic information, if provided at all, is typically conveyed through Group Objects.

By understanding the concepts of Group Objects, Group Addresses, parameters, and diagnostic properties, system integrators and installers can effectively configure and diagnose KNX devices to meet their specific requirements. The following table shows different Point Interface Types of Datapoints:

Point Interface Type	Description
	A communication interworking output Group Object (if.o) sends status events (e.g., switch ON/OFF) or periodic updates (e.g., temperature sensor) from a sender to one or many Receivers.
	A communication interworking input Group Object (if.i) receives events from one or many senders. An input Group Object receives either status events (e.g., light switch ON/OFF) or periodic updates from senders.
	Parameter (if.p) and diagnostic (if.d) properties are the basic interfaces of a Functional Block and define whether a client can only read or read and write the property value or metadata.

3.3 Transport Binding

Transport binding refers to connecting a higher-level protocol to a specific transport protocol (such as TCP, UDP, or CoAP) to facilitate data transmission across a network. The following section explains what

technologies and concepts KNX IoT utilizes for the Application Layer, Transport Layer, and Network Layer for seamless communication.

3.3.1 Application Layer

The KNX Point API primarily defines how the KNX application model, such as Volume 7 of the KNX Specification, is mapped to a REST data model. For KNX certification, the CoAP protocol on application layer must be supported for data transmission. The KNX IoT Point API is fully compatible with the CoAP standard, which simplifies integration with existing open-source libraries, as the KNX IoT Point API adheres strictly to the corresponding RFC specifications.

Alternatively, it is also possible to use other protocols, such as HTTP. This flexibility allows manufacturers to adopt the KNX data model while utilizing a different transport protocol. Notably, an existing RFC outlines how CoAP can be mapped to HTTP [RFC 8075]; however, this falls outside the scope of KNX IoT, as only CoAP can be KNX certified.

3.3.2 Network Layer

At the network layer, KNX IoT requires the use of IPv6, which offers several key advantages that streamline network management, that is important particularly in OT networks.

Automatic Address Configuration: IPv6 enables devices to autonomously generate addresses via Stateless Address Autoconfiguration (SLAAC), eliminating the need for manual setup or DHCP, minimizing network management, and supporting dynamic scalability in OT environments.

Enhanced Multicast Support: IPv6 improves multicast traffic handling, enabling efficient data transmission to multiple devices simultaneously. Features such as multicast listener discovery and its structured addressing system simplify group communication and reduce network traffic. This is particularly valuable in constrained networks like Thread, where the structured addressing is utilized to compress IPv6 addresses with 6LoWPAN [RFC 8138], optimizing performance and reducing network traffic.

End-to-End Transparency: IPv6's vast address space allows direct device-to-device communication, bypassing NATs, reducing complexity, and improving performance, cybersecurity, and troubleshooting. Its simplified header structure and flow label feature enhance router efficiency and quality of service.

3.3.3 Transport Layer

KNX IoT defines both UDP and TCP as transport layer protocols. However, UDP is mandatory for KNX certification, while TCP is optional. UDP, in combination with CoAP, is used for both device configuration and runtime interworking. TCP is utilized whenever the device category supports it. When used, TCP is used alongside TLS for device configuration or vertical integration. This is particularly important because it allows non-KNX certified third-party devices to use standard CoAP libraries with standard TLS, simplifying cybersecurity and the vertical integration of KNX IoT Point API devices.

3.3.4 Data Link Layer

In a Converged IP stack, the IP layer serves as an abstraction from the underlying Data Link Layer. Neither the client nor the server is aware of which Data Link Layer is being used by the communication partner. This abstraction makes the communication system highly flexible and allows for better scalability. For example, constrained Data Link Layers (e.g., Thread with 802.15.4) network segments can be interconnected via an unconstrained network backbone (Ethernet).

The Data Link Layer acts as the interface between the Physical Layer, which manages the electronics of a specific communication medium (e.g., wired or wireless), and the upper network layers. At this layer, the transmitted data, referred to as a "frame," undergoes a framing process defined by the Data Link Layer protocol during transmission. This involves organizing a sequence of bits in a format compatible with the sending electronics. Upon reception, the Data Link Layer interprets the incoming bitstream from the receiving electronics and decodes it into a frame suitable for forwarding to the upper network layers.

4 Network Topology

The following sections explain the runtime interaction models for horizontal interworking and vertical integration, using KNX network topology examples. Unlike vertical integration, which involves coordination between different hierarchical levels, horizontal interworking focuses on the interaction between similar devices operating at the same level.

4.1 KNX Network Topology

The KNX IoT system currently distinguishes between the KNX IoT Point API profile and KNX IoT Router profile that includes a KNXnet/IP interface.

KNX IoT Point API: A KNX IoT Point API device supports Group Object inputs and outputs that can be configured to exchange Group Value messages, enabling group communication from senders to receivers.

KNX IoT Router: The KNX IoT Router serves as a bridge, translating KNX IoT CBOR Group Value messages to KNXnet/IP messages and vice-versa. This facilitates the exchange of messages between different KNX areas via the KNX Backbone Line.

In a KNX system, IP is used as the KNX Backbone Line and for this purpose, KNXnet/IP routers are used. KNXnet/IP routers have an Ethernet interface on their "primary side" as well as a KNX TP or KNX IoT connection on the "secondary side" and forward KNX telegrams to other KNXnet/IP or KNX IoT Routers using the routing method. They make the topology of a KNX system more flexible by adding Ethernet as a transmission medium to the KNX system. KNXnet/IP routers can be used both as line couplers and as segment couplers and KNX routers are capable of filtering messages.

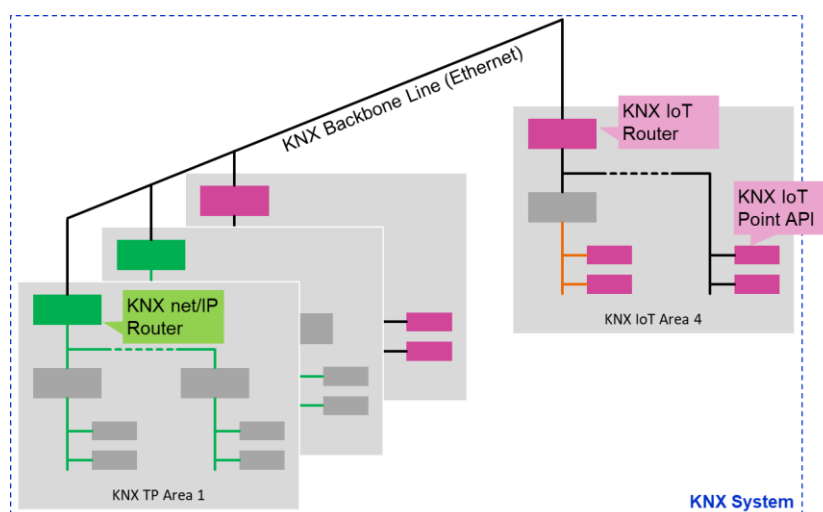


Figure 4-1: Coupling of KNX TP and KNX IoT areas with KNXnet/IP and KNX IoT Routers.

All KNXnet/IP routers also support tunnelling, enabling them to function as IP programming interfaces for other clients such as ETS. Furthermore, KNXnet/IP routers can be used to connect entire systems via Ethernet.

This is particularly advantageous in scenarios where separate KNX areas in two different buildings need to be integrated into a single unified system. If an Ethernet connection already exists between the buildings, as is often the case in commercial or institutional settings, no additional KNX specific cabling (TP1) is required, simplifying integration and reducing installation efforts.

4.2 Vertical Integration

Vertical integration in OT systems involves establishing seamless connectivity and integration across various Purdue model levels (see [PERA]) and cybersecurity zones. Within the context of HBES/BACS, vertical integration specifically refers to subsystem integration and interaction of building automation components across hierarchical levels. This includes the coordination and interconnection of diverse systems and functionalities, ranging from domain-specific control systems within the building, such as HVAC, lighting, and fire safety systems, to higher-level management and monitoring systems.

4.2.1 Subsystem Integration

The following example describes an EMS (Energy Management System) with a HEMS (Home Energy Management System) at its center. The HEMS manages various energy producers and consumers.

KNX IoT simplifies vertical integration by supporting open-source libraries like libCoAP, which supports cybersecurity features such as TLS out-of-the-box. This allows HEMS manufacturers (non-certified KNX IoT clients) to easily integrate KNX IoT Point API devices using open-source components. TLS simplifies credential management with a private/public key system. TLS is an optional feature in KNX IoT for devices with sufficient resources, while OSCORE application layer security, designed for also supporting constrained devices, is mandatory for all KNX IoT Point API devices and is used for horizontal interworking. KNX IoT also allows OSCORE with symmetric key cybersecurity for vertical integration of non-certified devices of course.

Since all Datapoints (Points) on a KNX IoT device are accessible via a REST API, a client can directly interact with them. Alternatively, the KNX group communication interface on a KNX IoT Router can be used, providing access to the data of configured Group Objects. The KNX IoT Router enables group communication with the entire KNX system. This also has the advantage of reducing the integration effort with single devices.

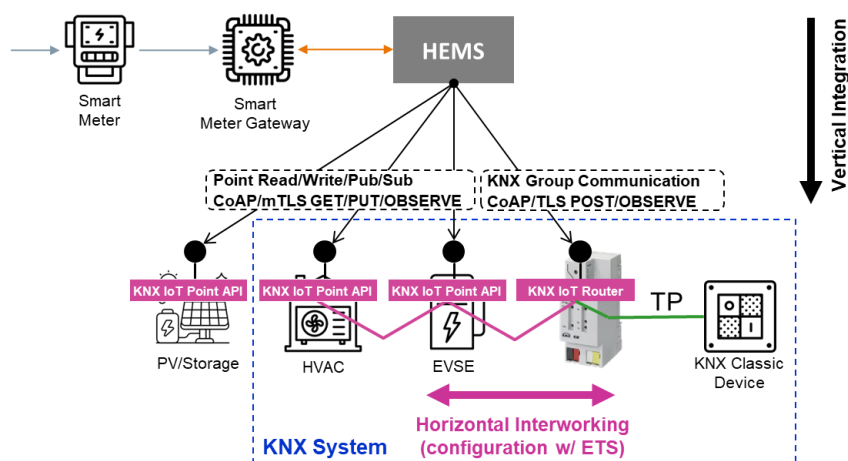


Figure 4-2: The figure depicts a PV/Storage device which is not directly connected to other KNX devices but is integrated into a HEMS via KNX IoT Point API. In this example, the HEMS manages both types of devices: KNX IoT Router with a KNX group communication interface and KNX IoT devices with the Point Read/Write/Pub/Sub interface (mutual authenticated TLS). Additionally, some KNX IoT devices are configured to enable horizontal interworking.

4.2.2 X.509 Certificate Management

X.509 certificate management is the process of issuing, installing, renewing, and revoking digital certificates. Digital identity, provided by a trusted certificate authority, remains one of the most robust and user-friendly solutions for authentication and encryption. An X.509 certificate establishes a binding between an identity (such as a device serial number, hostname, organization, or individual) and a public key (e.g., ECDSA, etc.) through a digital signature. Certificates can either be signed by a certificate authority or be self-signed, depending on the use case.

Manufacturers have the option to equip devices with TLS and a manufacturer-issued X.509 certificate, known as an LDevID [802.1AR]. This certificate enables a tool or service to authenticate the device and configure it with a new installation-specific LDevID [802.1AR] X.509 certificate. Another method involves triggering the device to obtain an LDevID from a Registrar via EST [RFC 7030] protocol, based on its IDevID.

Another more lightweight option a manufacturer can choose, especially in cases where no Public Key Infrastructure (PKI) is available to issue an IDevID, is to enroll the device using the standard KNX IoT enrollment mechanism based on a passphrase. In this case, a tool acts as the certificate authority, generating the LDevID for the devices and writing it to them in a subsequent step. The KNX IoT Point API specification outlines all three device profiles, but manufacturers have the flexibility to implement only a subset of them if needed.

The image below illustrates the lightweight option for configuring an LDevID on a KNX IoT device using ETS, followed by the setup of the corresponding access rights.

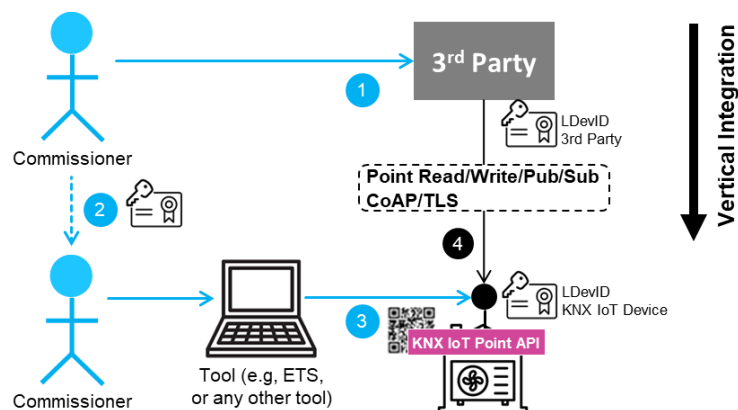


Figure 4-3: The figure shows the lightweight option with the standard KNX IoT configuration process where a commissioner configures the LDevID with a tool (1,2,3,4) or, as a second option, activate the hand-over process based on a passphrase (3,4,1). The runtime communication (4) between the third-party device and the KNX IoT device uses a mutual authenticated TLS session.

In the following, each step that the two commissioning engineers need to perform to configure the LDevID are described in more detail:

Retrieve LDevID (1): In the first step, the Commissioner 1 of the third-party device (e.g. the HEMS) retrieves the X.509 LDevID certificate with the public key. If necessary, he may need a tool for this, or the device may have a web interface where the certificate is made available.

Hand-over LDevID (2): The LDevID is then passed to the KNX IoT Device Commissioner 2, who configures the access rights for the third-party device and the LDevID certificate on the corresponding device.

Configure & Download (3): After configuring the access rights and LDevID, the configuration set is written to the device. After the download, the third-party device has access to the Datapoints with the corresponding access rights.

Connect & Accept (4): Once the third-party device has been configured to trust the KNX IoT device, for example, based on QR-Code information, the third-party device can discover and start reading or writing Datapoints. The third-party device starts a mutual authenticated TLS session. With CoAP, it is also possible to subscribe to Datapoints, and when there is a change, the new information is automatically sent from the KNX IoT device to the third-party device via secured TLS session.

4.3 Horizontal Interworking

Horizontal interworking involves seamless collaboration and communication among devices at the same functional level. Unlike vertical integration, which typically involves coordination between different hierarchical levels or functions, horizontal interworking focuses on the interaction between similar devices operating at the same level.

4.3.1 Device-to-Device Group Communication

Device-to-device group communication is a mechanism that allows one device to send real-time data to another device when a specific event occurs. Instead of the receiving device repeatedly polling for updates, the sending device automatically sends a notification or payload to a predefined endpoint using the CoAP protocol. Essentially, a device-to-device group communication acts as a lightweight, event-driven communication method, enabling devices to exchange information efficiently.

In the context of device-to-device group communication, data exchange can occur either through multicast or unicast communication, depending on the use case:

Multicast Message Exchange: In multicast communication, a CoAP message is sent to multiple devices simultaneously using a IPv6 multicast address. This method is particularly useful for scenarios where the same message or event notification needs to be distributed to a group of devices at once. For example, a device can use Multicast Message Exchange to send a notification to a group of KNX IoT devices within a network, such as turning on multiple lights with a single message.

Direct Unicast Message Exchange: In unicast communication, a CoAP message is sent directly from one device to another using the specific IPv6 unicast address of the recipient. This is ideal for one-to-one communication where reliable and confirmed data exchange is required. For instance, a Direct Unicast Message Exchange could be used to command a specific KNX IoT device to perform an action, such as adjusting the setpoint on a single actuator.

The image below illustrates the system architecture of a KNX IoT system, which is seamlessly connected to a KNX Classic system via a KNX IoT Router. The configuration of both KNX IoT Point API devices and KNX Classic devices is managed using ETS (Engineering Tool Software). In the example, ETS configures device-to-device group communication across the system. This configuration (horizontal interworking) allows devices to send KNX group communication IPv6 messages directly to the recipient via either multicast or direct unicast device-to-device group communication. ETS is a manufacturer-independent configuration tool specifically designed for planning, designing, and configuring KNX installations, ensuring interoperability and efficient system setup.

For the Thread subsystem, a DCA (Device Configuration App in ETS) is used to configure the Thread Border Router in the following example. Specifically, the DCA manages the allow-list of Thread devices permitted to

join the Thread network. While the KNX system is configured using ETS, the Thread subsystem is configured separately using a DCA.

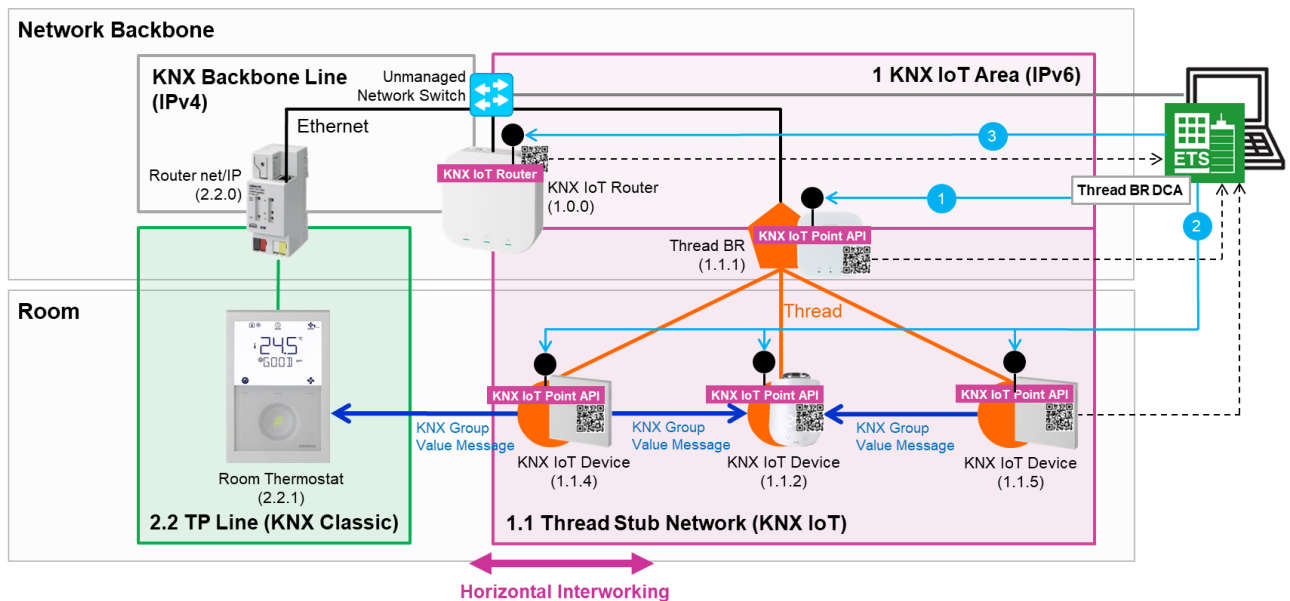


Figure 4-4: The figure illustrates step-by-step how devices are configured within a KNX IoT system. In the first step, the DMC or QR code is scanned to retrieve the passphrase and serial number. This information is used to onboard the devices into a network (if necessary). Once the devices are available on the network, the corresponding device configuration is written to the devices. As the final step, the KNX IoT Router is configured to ensure that Group Value messages from the KNX Classic zone are transmitted to the KNX IoT zone or vice versa (firewall). The KNX IoT Router can function as an independent standalone device or be integrated with KNXnet/IP Router hardware.

4.3.1.1 ETS Network Topology View

In ETS, a KNX system can be organized into distinct network zones. One example is the division between a KNX Classic zone and a KNX IoT zone, as depicted in the following picture. The KNX IoT zone can further be subdivided, such as by creating a grouping of Thread devices within a Thread stub network. By utilizing the Thread BR DCA, one can configure an allow-list to enable devices to join the Thread network.

This configuration of network zones serves as the foundation for horizontal interworking and network segmentation, facilitating the establishment of distinct cybersecurity zones.

Topology

Topology Backbone

1 KNX IoT Zone 1

1.1 Thread stub network 1

1.1.1 OCT110.BR Thread Border Router

1.1.2 SSA911.02TH Wireless Valve Actuator

1.1.4 QPA2892/WI Wireless Room Sensor (T+rH+CO2)

1.1.5 QAA2890/WI Wireless Room Sensor (T)

1.2 Thread stub network 2

1.0.0 OCT200.KNR IoT Router

2 KNX Classic Zone 2

2.2 TP line

2.2.0 IP Router Secure N 146/03

2.2.1 RDG200KN Room Thermostat

Group Objects0ParametersDCA

1.1.1 Thread Border Router > DCA > Allow List Configuration

Network Status

Allow List Configuration

+ Add- DeleteCopy To Allow List Parameters

Address	Product	Order Number	Serial Number	EUI-64	Select All
1.1.2	SSA911.02TH Wireless Valve Actuator	SSA911.02TH	00FD10F3A7B4	0080E12601678266	☐
1.1.4	QPA2892/WI Wireless Room Sensor (T+rH+CO2)	QPA2892/WI	00FD10F1C8E4	0080E12601677299	☐
1.1.5	QAA2890/WI Wireless Room Sensor (T)	QAA2890/WI	00FD10F1D6F0	0080E126016CEE8	☐

Figure 4-5: The ETS network view shows a KNX IoT line and a KNX TP line.

4.3.2 KNX IoT Router Configurations

The KNX IoT Router plays a key role in integrating KNX subsystems or connecting a third-party system to a KNX system. The KNX IoT Router can serve as a Vertical Integration endpoint, allowing a third-party device to either write to configured Group Objects (GOs) or receive notifications triggered by events in the KNX Classic system or the KNX IoT system.

To enhance flexibility and network security, device manufacturers may implement two separate Ethernet interfaces on the KNX IoT Router: one dedicated to Vertical Integration and the other for KNX Horizontal Interworking (such as the KNX Backbone or exclusively the KNX IoT Area). This physical separation of networks ensures the isolation of KNX devices from other network segments.

The integration of a KNX IoT system into a KNX Classic system is also achieved through the KNX IoT Router. There are several methods to configure the KNX IoT Router. One approach is to use ETS to set up KNX IoT devices for device-to-device group communication, either via unicast or multicast, as previously described. For unicast, the KNX IoT Router must be configured as an endpoint in the Function Point Table. For multicast, the KNX IoT Router must be placed within the corresponding multicast zone. Alternatively, the KNX IoT Router can be configured using Endpoint Subscription Message Exchange.

Endpoint Subscription Message Exchange: With Endpoint Subscription Message Exchange, the KNX IoT Router sends a CoAP Observe message to the configured KNX IoT device to subscribe to the KNX group communication endpoint (/k). Once successfully registered, the KNX IoT device sends back all notifications for which the appropriate permission rights are granted on the KNX IoT device. The KNX IoT Router then publish Group Value messages (KNX group communication) to other KNX IoT devices or KNX Classic devices. This approach is ideal for scenarios requiring reliable and confirmed data exchange. Additionally, it can simplify the integration process by decoupling the configuration of KNX IoT device-to-device communication from the KNX Classic or third-party system integration.

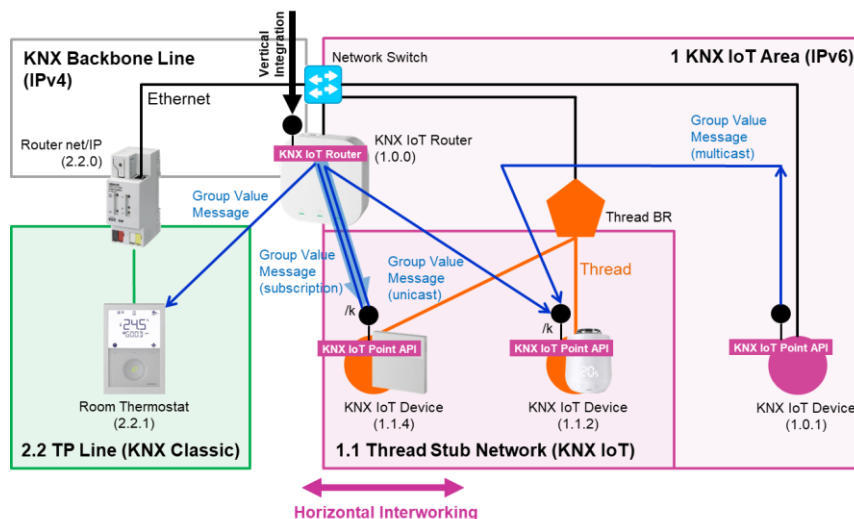


Figure 4-6: The image shows a KNX IoT system exchanging data via a KNX IoT Router that is configured with Endpoint Subscription Message Exchange (Individual Address 1.1.4). On the other hand, the KNX IoT device with the Individual Address 1.0.1 is configured to send messages using Multicast Message Exchange. The device with Individual Address 1.1.2 gets updates directly via Multicast Message Exchange from device 1.0.1 and indirectly via KNX IoT Router from the device with the Individual Address 1.1.4 with Direct Unicast Message Exchange. The KNX IoT Router uses KNX net/IP routing for group communication (Group Value message) with the KNX Classic system.

5 Application Engineering

An HBES/BACS system is a system consisting of control devices, processing equipment, network interfaces, and gateways, where the functions are distributed and linked through a common communication and data model, managing multiple applications in home and building premises (see also [IEC 63044-1]). Examples of applications are heating, alarming, shading and lighting.

The following sections explain how the functionalities of devices in a distributed KNX system can be seamlessly interconnected using input and output Group Objects (GO). These links and the corresponding permissions are established through the configuration of the Function Point Tables on KNX IoT devices by the ETS software.

5.1 Application Functions

The concept of an Application Function is fundamental in building automation systems, as it encapsulates the required functionality based on customer needs. An Application Function represents, for example, the lighting control functionality or room heating control functionality within a space, achieved through the interaction and coordination of one or multiple devices. For example, when a light switch (sensor device) turns on a lamp (actuator device) in a room, the resulting behavior represents a lighting control Application Function of this room.

Devices that participate in an Application Function process input data from other participating devices like sensors, actuators, or command messages of devices belonging to other Application Functions. They can also produce output data, which can be utilized by actuators or downstream Application Functions to manage specific process variables or states within a control loop. How data is exchanged between devices within an Application Function, depends on the Application Function itself and the specific functionality that it implements. Depending on these requirements, one device interworking pattern with the corresponding message exchange configuration in the Function Point Table may be more suitable than another.

In the following sections, first device interworking patterns are introduced with a simple fan coil unit and a light control example. Subsequently, these and additional device interworking patterns are explained in more detail.

5.1.1 Simple Fan Coil Example (Heating only, Temperature Driven)

The following example demonstrates how the device interworking pattern “Confirmed 1:1 Device Communication” and “Controller/Peripheral Subscription Communication”, which are explained in the following sections, can be combined within a single application function. The objective in this example is to minimize configuration and centralize the device discovery on the controller.

A two-pipe fan coil unit (FCU) is a component commonly used in building automation systems for heating applications. Its primary function in this application function is to control the temperature of a specific area within a building. The following plant diagram, as outlined in KNX Volume [7/13/00], illustrates a typical two-pipe FCU setup with the corresponding functional blocks.

The diagram includes a combined fan and heating coil actuator along with a room sensor. The fan is responsible for circulating air within the space and can operate at different speeds to adjust airflow as needed. The control of the two-pipe FCU is influenced by several inputs, such as temperature sensors (room and duct), window sensors, and time schedules. Users can also adjust the setpoint via a room unit (HMI). These inputs are processed by the Room Setpoint Manager, which works in conjunction with the Fan Coil

Control controller to ensure precise operation of the FCU. This allows the system to maintain optimal comfort and energy efficiency within the room. These communications use Controller/Peripheral Subscription Communication.

The two-pipe FCU is managed by a time schedule program. The time schedule program is assigned to the Plant Management Control Application Function. Since Plant Management Control is a higher-level function, the new value from the time schedule program is written to the Room Setpoint Manager using the Confirmed 1:1 Device Communication interworking pattern.

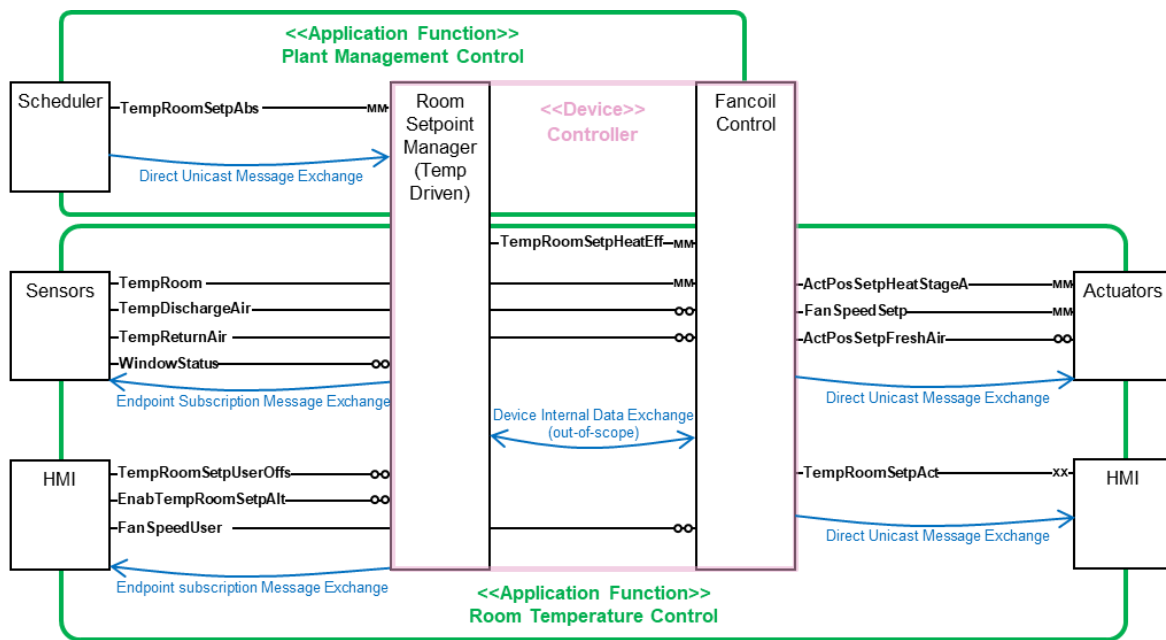


Figure 5-1: MM indicates a mandatory connection, while OO represents an optional connection. XX stands for diagnostic data. Additionally, the image illustrates the types of message exchanges that can be applied between the various functional blocks. The Room Setpoint Manager and Fan Coil Control functional block are in the same device.

5.1.2 Light Control Example

The following example illustrates how the device interworking patterns "Low-latency N:N Peripheral Communication" and "Confirmed 1:1 Device Communication," detailed in the subsequent sections, can be seamlessly integrated within a light control application. The primary goal of this example is to minimize communication between the switch and the light actuator. Additionally, to ensure all light actuators switch on or off simultaneously, a multicast messaging approach is employed. Meanwhile, to maintain accurate status updates on the HMI, a reliable Confirmed 1:1 Device Communication is utilized.

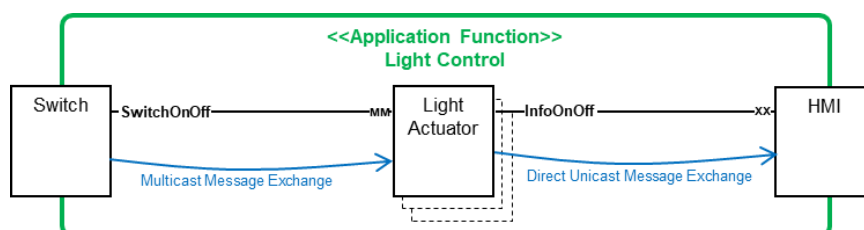


Figure 5-2: The figure shows a **Multicast Message Exchange** example between a light switch and light actuator. The current light actuator status is visualized on an HMI using **Direct Unicast Message Exchange**.

5.2 Device Interworking Patterns

The following chapters present various device interworking patterns, as demonstrated in the previous application function example and available in KNX IoT. These patterns are accompanied by practical examples to provide context and clear guidance on effectively utilizing the different types of message exchange.

5.2.1 Confirmed 1:1 Device Communication

Confirmed 1:1 Device Communication is used when a reliable exchange of information is required. This means that if a message is not acknowledged by the recipient, it can be resent by the sender. Confirmed 1:1 Device Communication utilizes the Direct Unicast Message Exchange type, requiring the IPv6 address to first be resolved based on the destination Individual Address (IA). This IPv6 address is stored by the initiator, so it does not need to be resolved again for subsequent communications unless the recipient device changes its IPv6 address. A prerequisite for the Direct Unicast Message Exchange type is that mDNS or multicast CoAP discovery must be functional within the corresponding network segment.

In the following example, after resolving the IPv6 address, the sensor first sends its value to the controller device. The controller processes this information and subsequently sends a new actuator command to the corresponding device.

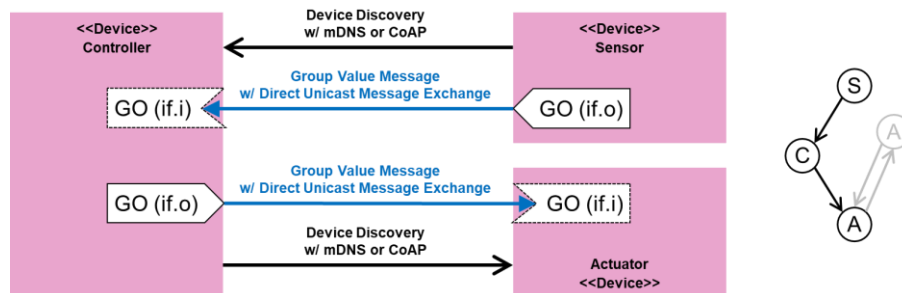


Figure 5-3: The figure illustrates bidirectional **Direct Unicast Message Exchange** example containing a controller device and peripheral devices with either sensor or actuator functional blocks.

5.2.2 Controller/Peripheral Subscription Communication

Controller/Peripheral Subscription Communication is used, for example, when a controller device requires sensor data or status information from an actuator. Controller/Peripheral Subscription Communication uses the Endpoint Subscription Message Exchange.

In the following example, the controller device subscribes to the KNX group communication endpoint (/k) of both the sensor and the actuator device (through a Direct Unicast Message Exchange). Subsequently, the actuator and sensor notify the controller whenever there is a change in the GO (General Object) through Endpoint Subscription Message Exchange. The controller processes this information and then sends a new actuator command to the corresponding device using a Direct Unicast Message Exchange (not in the example). One advantage of this design is that device discovery is initiated by the controller device, resulting in a star-shaped communication pattern with the controller at its center.

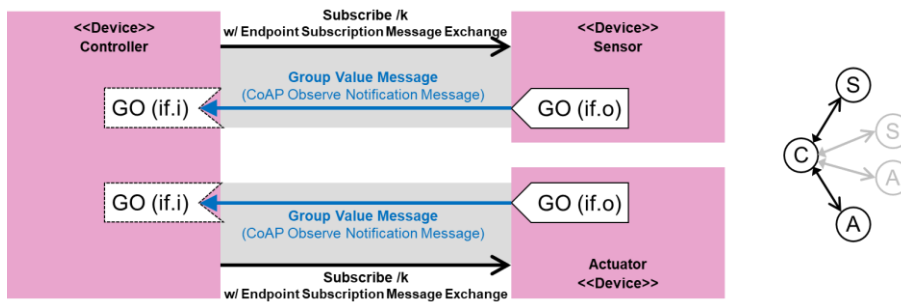


Figure 5-4: The figure illustrates **Endpoint Subscription Message Exchange** example containing a controller device and peripheral devices with either sensor or actuator functional blocks. The controller device initiates the first subscription request (CoAP Observe). This creates a star-shaped communication pattern involving a controller (C), a sensor (S), and an actuator (A), as illustrated on the right-hand side. The resulting Group Value Message are sent to the subscriber (CoAP Observe Notification Messages).

5.2.3 Low-latency N:N Peripheral Communication

Low-latency N:N Peripheral Communication utilizes the Multicast Message Exchange type. This means that a single message can be used to control multiple actuators simultaneously. Another advantage is the simpler configuration, for instance, when multiple light switches and lamps (actuators) at the same time are used in an application function. However, the disadvantage of this type of message exchange is that the messages are not confirmed.

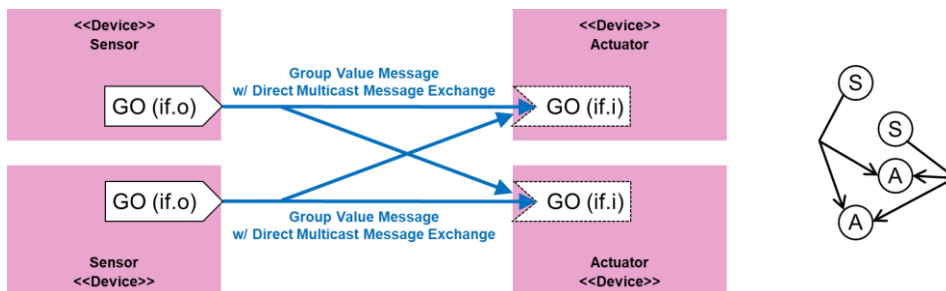


Figure 5-5: The figure depicts **Multicast Message Exchange** example involving multiple devices with sensor functional blocks and multiple devices with actuator functional blocks. A single message is used to simultaneously command multiple actuators.

6 Device Configuration

The following chapters provide a detailed explanation of the Function Point tables, the Access Control List and the corresponding message exchange types, addressing readers who seek a deeper understanding of the specific device configurations.

The permissions for Group Objects and Datapoints are configured in the Access Control List, detailed in a separate section. However, access token examples are already provided alongside the Function Point Tables for better clarity and context.

6.1 Function Point Tables

This chapter provides a detailed explanation of the Function Point tables, addressing readers who seek a deeper understanding of the specific configurations required by the tool.

The Function Point Tables of KNX IoT Point API device are a configuration mechanism that links KNX Group Objects to specific Datapoints within a device. Function Point Tables consist of three tables: the Group Object

Table (fp/g), the Function Point Recipient Table (fp/r), and the Function Point Publisher Table (fp/p). These tables are described in more detail because they represent a key concept, and it is essential to understand their interconnections and how information from various sources contributes to the resulting Group Value message.

The Group Object Table contains Group Object Table Items (or Group Object Items in some drawings) that map Group Addresses to Datapoints and define their behavior and communication capabilities (e.g., read/write/transmit, etc.). The Function Point Recipient Table contains a list of recipients with the configured endpoints for outgoing group messages. The Function Point Publisher Table lists the source from which incoming group messages are received.

Together, these tables enable Multicast Message Exchange, Direct Unicast Message Exchange, and Endpoint Subscription Message Exchange for runtime interworking in KNX installations. ETS configures these tables during commissioning, assigns Group Addresses, and sets up communication endpoints (see also section “Group Communication”). The following sections include examples that illustrate how the tables are configured for each type of message exchange.

6.1.1 Multicast Message Exchange

In the following example, the temperature value, for instance when a value change occurs, is sent from the device with the Individual Address 5678 (=1.6.46) to the device with the Individual Address 5699 using a multicast IPv6 message.

Function Point Recipient Item: On the sender side, the Datapoint is associated with the Group Object Table Item configured with the same href (resource path). This Group Object Table Item is then connected to one or more Function Point Recipient Items that have the same Group Address (ga=1234). The Group Address is also used to find the corresponding Access Token Item (scope=Group Address), which contains the OSCORE credentials used to encrypt the Group Value Message.

Multicast Group Value Message: The IPv6 multicast address is composed of two parts: the first part is fixed and includes the IPv6 multicast prefix and the KNX system installation ID, while the second part is derived from the group ID (grpid) specified in the Function Point Recipient Item. The grpid is configured by ETS and can represent a single Function Point or be used to transmit multiple Function Points via the same IPv6 multicast address. The Group Value Message includes the sia (source Individual Address), the service type (st), the Group Address (ga), and the value from the sender's Datapoint.

Function Point Publisher Item: On the receiving side, the corresponding OSCORE credentials (such as the OSCORE recipient context) are extracted from the Multicast Group Value Message to identify and locate the associated Access Token (kid=71) within the Access Control List. If the credential (AES key derived from a pre-shared symmetric secret) corresponds with the configured settings in the Access Token Item, it is utilized to decrypt the message. Following decryption, access rights are verified by ensuring that the scope attribute includes the corresponding Group Address. If all validation checks are successful, the value is written to the Datapoint associated with the Group Object Table Items that matches the configured Group Address (with the same href).

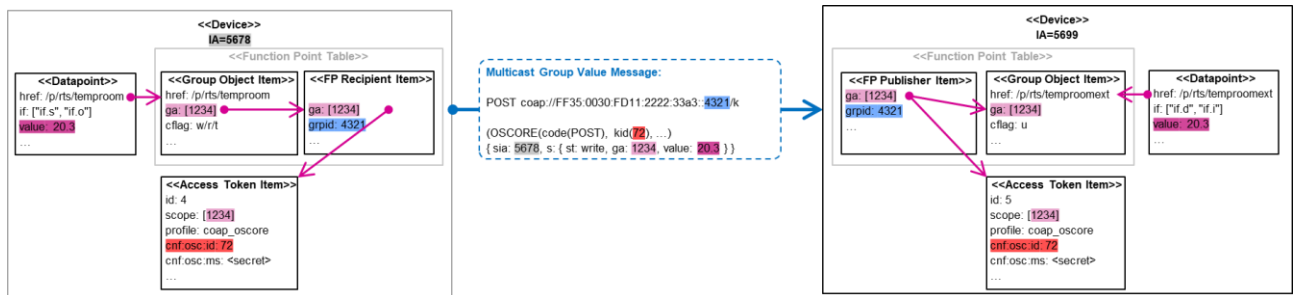


Figure 6-1: The figure illustrates two KNX IoT Point API devices exchanging data through an IPv6 multicast request. The colors represent the source or destination of the information that is ultimately used in the CoAP message.

6.1.2 Direct Unicast Message Exchange

In the next example, the temperature value is transmitted using a unicast IPv6 message again from the device with Individual Address 5678 to the device with Individual Address 5699.

Function Point Recipient Item: Just like with Multicast Message Exchange, the Datapoint is linked to the Group Object having the same href. The connections between the Function Point Recipient Item and the Group Object Table Item remain identical for both unicast and multicast configurations. However, for unicast message exchange, at (at=3) is used to link the Function Point Recipient Item with the Access Token. In addition, the Function Point Recipient Item includes the recipient's Individual Address (ia=5699). This means that during the initial transmission or if the recipient's IPv6 address changes, the sending device must first resolve the IPv6 address using mDNS or CoAP. Once resolved, the IPv6 address is locally stored along with the Individual Address (ia) for future transmissions in the IPv6 Item (see drawing below).

Unicast Group Value Message: The IPv6 unicast address from the IPv6 Item is used as the destination address. The payload of the Group Value Message contains the same information as the previous multicast example.

Function Point Publisher Item: With Direct Unicast Message Exchange, a Function Point Publisher Item is not required on the receiving side. If the OSCORE key id (kid) matches an access token, then it is used to decrypt the message and validate group address access rights (scope=[4321]). Once all checks are successfully completed, the value is written to the Datapoint associated with the Group Object Table Item that has the matching href.

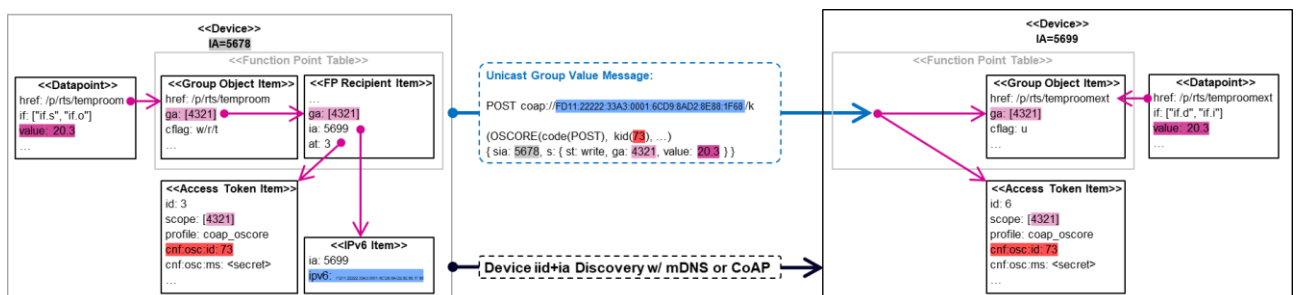


Figure 6-2: The figure illustrates two KNX IoT Point API devices exchanging data through an IPv6 unicast request. The device with the Individual Address 5678 discovers the device with the corresponding Individual address (ia) that belongs to the same KNX installation (iid). The colors represent the source or destination of the information that is ultimately used in the CoAP message.

6.1.3 Endpoint Subscription Message Exchange

The Function Point Publisher Table is located on the Group Value Message receiving side for Multicast Message Exchanges. Similarly, this applies to Endpoint Subscription Message Exchanges where the Function Point Publisher Table contains the configuration from where to receive Group Value Messages. Hence, the initiator of the CoAP subscription operates on the Group Value Message receiver side (ia=5699).

In this scenario, the receiver, such as a controller Point API device or a KNX IoT Router, sends a CoAP Observe request (subscription) to the Group Value Message sender (ia=5678), as defined in the Function Point Publisher Item. The publisher then reacts by sending the group object Datapoint values configured on its side back to the initiator via a CoAP notification each time a point change occurs.

Function Point Publisher Item (subscription): The initiator device, with an Individual Address configured in the Function Point Publisher Item (ia=5676), resolves the IPv6 address of the Group Value Message sender using mDNS or CoAP, if the address has not already been resolved and stored in the IPv6 Item.

CoAP Observe Unicast Message: Once the IPv6 address is successfully resolved, the initiator device subscribes to the KNX group communication endpoint (/k) on the Group Value Message sender (ia=5678) by sending a CoAP Observe Unicast Message request. This request is encrypted using the OSCORE credentials from the Access Token (at=6), which is referenced in the Function Point Publisher Item.

Access Token Item: On the receiving side, the Access Control List is checked to verify if there is an Access Token Item containing the corresponding OSCORE credentials (cnf.osc.id=kid). If the kid from the CoAP message matches, the secret (pre-shared symmetric key) associated with the Access Token Item is used to decrypt the message and validate the permissions in the scope attribute.

CoAP Notification Unicast Group Value Message: The Group Value message sender then responds by sending back all configured group objects to the subscriber for instance when a value change occurs. However, only changes for Datapoints that have associated Group Object Table Items where the Transmit Flag is set and the sending Group Address is contained in the scope of the subscription Access Token, are sent. In the example, the access scope is configured with if.g.s, defining that all configured Group Objects are transmitted to the corresponding device.

Function Point Publisher Item (notification): On the receiving side, the access scope defined in the Access Token, as referenced in the Function Point Publisher Item, is also set to if.g.s, allowing access to all Group Addresses. However, values are written to the corresponding Datapoints only if the Group Address is explicitly configured in Function Point Publisher Items and the corresponding Group Object Table Items (ga=1234).

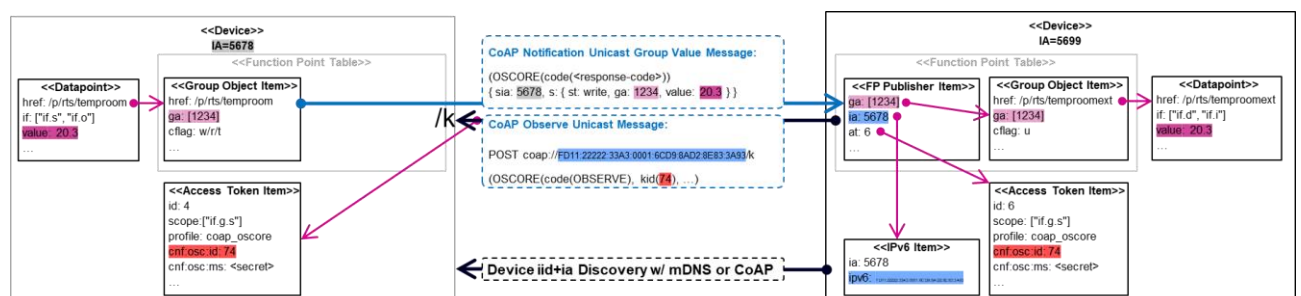


Figure 6-3: The image illustrates a KNX IoT Point API device controller (IA=5699), with an outgoing CoAP Observe request (CoAP GET + CoAP Observe option), and an incoming Group Value unicast notification from a KNX IoT Point API device with the Individual Address 5678. The colors represent the source or destination of information that is ultimately included in a CoAP message.

6.1.3.1 KNX IoT Router Endpoint Subscription Message Exchange

The next example illustrates again an Endpoint Subscription Message Exchange, this time for a KNX IoT Router (see also section “KNX IoT Router Configurations”). In this case, the KNX IoT Router is configured to subscribe the KNX group communication endpoint (/k) on other KNX IoT Point API devices. However, instead of using the Group Object Table, the message is processed through the Group Address Mapping Table and is subsequently forwarded to the KNX Classic System. The Group Address Mapping Table converts the CBOR message value payload of a specific Datapoint Type (“t”) into a KNX Classic message. Since this message was received as a unicast message, the Function Point Recipient Table is additionally checked to determine whether the message needs to be forwarded to other KNX IoT devices (not shown in the picture).

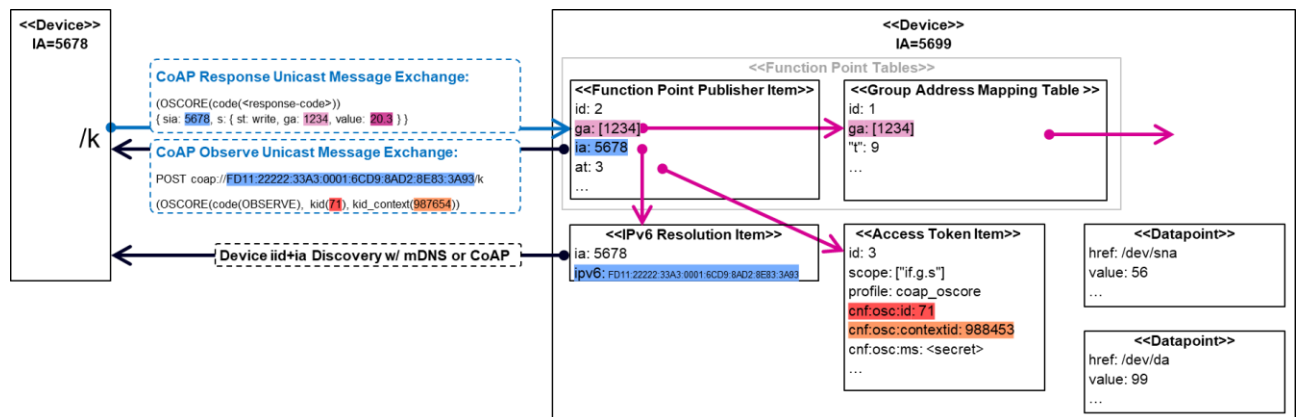


Figure 6-4: The image illustrates a KNX IoT Router, with an outgoing CoAP Observe request, and an incoming Group Value unicast notification from a KNX IoT Point API device with the Individual Address 5678. The incoming notification goes via Group Address Mapping Table into the KNX Classic system.

6.1.3.2 3rd Party Endpoint Subscription Message Exchange

In the following example, a non-KNX client subscribes to the KNX group communication endpoint (/k) on a KNX IoT Router (see also section “KNX IoT Router Configurations”). The KNX group communication endpoint (/k) and the corresponding Access Token are configured, enabling the third-party client (vertical integration) to receive all messages transmitted through the KNX IoT Router. This means that all Group Value messages, for which the third-party client has the appropriate permissions configured in the access token, are forwarded to the third-party client. An access token has been configured on the KNX IoT Router, which includes an X.509 certificate configured in its trust list (auth/crts). This certificate is utilized for authentication and authorizes the third-party client to access all Group Value messages (scope=if.g.s) configured in the access token. These messages can originate from either the KNX Classic system or the KNX IoT system. With if.g.s the third-party client receives all messages configured in the Group Address Mapping Table. The scope can also be defined with group addresses to selectively send Group Value messages to the third-party client (commonly referred to as security proxy). Of course, the third-party client can also send Group Value messages to the KNX IoT Router using Direct Unicast Message Exchange.

Access Scope	Description
if.swu	Permission to update the software.

In the following example, a client, either a KNX IoT device or non-KNX device, accesses a CoAP resource directly (e.g., vertical integration). On the device with the Individual Address 5699, an access token was configured along with the corresponding OSCORE credentials. This means that the client, which also possesses the same OSCORE credential, is authorized to access all Datapoints with the access scope "if.d". In this particular case, for example, the Datapoint with the href "p/rts/temproom".

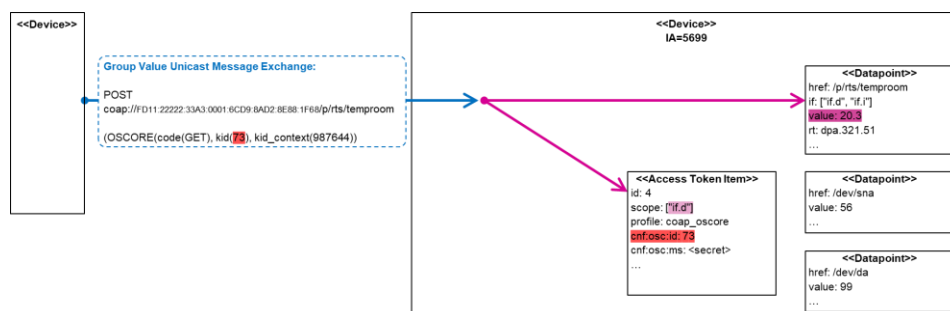


Figure 6-6: The image illustrates a device accessing directly a CoAP resource on a KNX IoT device.

7 Conclusion

The KNX IoT Point API provides a standardized framework for runtime interworking, facilitating efficient communication through various message exchange patterns, including Multicast, Direct Unicast, and Endpoint Subscription Message Exchange. These patterns, configured via ETS, ensure flexibility and reliability in device-to-device and system-to-system communication. Furthermore, the use of Function Point Tables and Access Control Lists enhances cybersecurity and simplifies the configuration of permissions, ensuring secure and controlled access to Datapoints.

KNX IoT also addresses emerging challenges in building automation, such as energy efficiency, cybersecurity, and the integration of IT and OT systems. By reusing existing infrastructure, supporting open-source libraries, and enabling seamless device discovery, KNX IoT minimizes installation complexity while providing robust tools for system integrators, installers, and manufacturers.

With its innovative approach to semantic data modeling, advanced runtime interworking capabilities, and support for both wired and wireless IPv6 communication technologies, KNX IoT is well-positioned to meet the demands of modern building automation systems. As the ecosystem continues to evolve, future ETS versions and expanded functionalities will further enhance its capabilities, ensuring KNX remains a leading solution in the HBES/BACS domain.

By adopting KNX IoT, device manufacturers, system integrators and building owners can unlock new opportunities for efficient system design, streamlined operations, and enhanced interoperability, ultimately driving innovation and sustainability in the building automation industry.

8 References

- [7/13/00] Volume 7/13/00, "Terminal Unit Functional Blocks-Interworking of Functional Blocks", KNXA.
- [802.1AR] IEEE Std 802.1AR-2018, "IEEE Standard for Local and metropolitan area networks–Secure Device Identity", IEEE, 2018.
- [EN50090-6-2] EN50090-6-2, "Volume 3/10/3 - KNX Information Model", <https://schema.knx.org/>.

- [EN50090-6-3] EN50090-6-3, "Volume 3/10/4 - KNX IoT 3rd Party API", <https://schema.knx.org/>.
- [EN50090-6-4] EN50090-6-4, "Volume 3/10/5 - KNX IoT Point API", <https://schema.knx.org/>.
- [IEC 63044-1] IEC 63044-1, "Home and Building Electronic Systems (HBES) and Building Automation and Control Systems (BACS) - Part 1: General requirements", <https://webstore.iec.ch/en/publication/32244>, 2017.
- [IEEE S802.3cg] IEEE 802.3cg-2019, "IEEE Standard for Ethernet - Amendment 5: Physical Layer Specifications and Management Parameters for 10 Mb/s Operation and Associated Power Delivery over a Single Balanced Pair of Conductors", 2019
- [PERA] Purdue Enterprise Reference Architecture, Wikipedia, September 2024, https://en.wikipedia.org/wiki/Purdue_Enterprise_Reference_Architecture.
- [Point API Schema] KNX IoT Point API Scheme, GITLab, February 2025, <https://gitlab.knx.org/public-projects/knx-iot-point-api-schema>.
- [RFC 4492] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 4492, IETF, August 2008.
- [RFC 6762] S. Cheshire, M. Krochmal, "Multicast DNS", RFC 6762, IETF, February 2013.
- [RFC 7030] M. Pritikin, P. Yee, D. Harkins, "Enrollment over Secure Transport", RFC 7030, IETF, October 2013.
- [RFC 7252] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, IETF, June 2014.
- [RFC 8075] A. Castellani, S. Loreto, A. Rahmanm, T. Fossati, E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, IETF, February 2017.
- [RFC 8138] P. Thubert, C. Bormann, L. Toutain, R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, IETF, April 2017.
- [RFC 8613] G. Selander, J. Mattsson, F. Palombini, L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, IETF, July 2019.
- [RFC 8949] C. Bormann, P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 8949, IETF, December 2020.
- [RFC 8259] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 8259, IETF, December 2017.
- [RFC 9383] T. Taubert, C. A. Wood, "SPAKE2+, an Augmented Password-Authenticated Key Exchange (PAKE) Protocol", RFC 9383, IETF, September 2023.