Thread and KNX IoT for Scalable, Secure, Interoperable Smart Buildings.

Bridging KNX and IoT for Seamless Interoperability and Advanced Networking

This Thread Technical white paper is provided for reference purposes only. The full technical specification is available publicly. To gain access, please follow this link: https://www.threadgroup.org/ThreadSpec.

If there are questions or comments on this technical paper, please send them to help@threadgroup.org.

Introduction

The primary objective of this white paper is to guide KNX Members and IoT professionals in developing KNX IoT devices based on Thread. The development process follows the established KNX device engineering methodology while leveraging modern physical and network layer technologies. At the same time, it preserves the fundamental KNX principles of vendor independence, interoperability, and seamless application-level integration.

In this paper, we outline the challenge of scaling KNX into modern IP-based wireless networks and present Thread as a robust IPv6-based mesh option under KNX IoT. We explain how KNX IoT coexists with KNX via ETS, the official configuration tool maintained by the KNX Association, and a KNX IoT Router. We also cover key enabling technologies, including the Thread Border Router and Thread 1.4 enhancements such as over-infrastructure support, improved diagnostics, and simplified commissioning. We also explore real-world examples and available system components to show feasibility, and a step-by-step guide to KNX Implementation (Chapter 3).

The audience for this white paper includes:

- KNX Members and manufacturers developing devices and appliances for B2B markets, installers, or integrators commissioning systems using commercially available tools like ETS.
- IoT system integrators and smart building solution providers delivering connected lighting, HVAC, energy management, security, and BMS projects using ETS.
- IoT development and application experts building KNX-compatible firmware, gateways, and services that leverage the KNX Information Model (KIM) and IP connectivity.

The content provides valuable insights for anyone invested in advancing IoT integration within smart buildings while maintaining KNX's trusted interoperability and vendor independence.

Authors:
Oskar Camenzind & Levy Toredjo

1.1 Background

The rapid evolution of the IoT has reshaped the landscape of smart homes and residential and commercial buildings. Consumer expectations of modern IoT devices demand that manufacturers bring the latest features and functionality to their offerings. For developers, this requires solutions that offer seamless interoperability, robust networking, and global scalability. An open-source stack is also increasingly important, giving developers greater visibility into the technology they are building on, more flexibility to adapt it to their needs, and access to a broader community contributing to its ongoing development.

KNX, a leading standard in smart home and building automation, is evolving to meet these demands through its IoT-focused extensions. By integrating Thread, a low-power, IPv6-based mesh networking protocol, this evolution to KNX IoT expands its capabilities to include IP-based wireless solutions.

This innovation also makes it possible for KNX IoT devices to operate on global frequency bands while maintaining KNX's core values of vendor independence, application-level interoperability, and seamless integration with the ETS (Engineering Tool Software). The move to KNX IoT aligns with the broader industry vision of creating scalable, secure, and energy-efficient networks that facilitate retrofitting and global deployment in the smart home and building market

1.2 Problem Statement

Manufacturers face significant challenges in meeting the demands of modern IoT markets. Many prefer wireless technologies that cater to global markets instead of regional, frequency-band-based solutions due to economies of scale. Moreover, the demand for retrofitting existing buildings with smart technologies remains unmet because most KNX-certified products rely on twisted-pair wiring, which requires new construction or major renovations.

IPv6, combined with mesh networking, offers a robust solution by enabling reliable, low-power, and scalable connectivity in smart homes and buildings. Thread supports these capabilities, ensuring secure device-to-device communication and broad interoperability across IoT ecosystems.

1.3 Importance

The adoption of KNX IoT with Thread technology is a pivotal step in addressing major global trends and market needs. Retrofitting opportunities in existing buildings represent a significant untapped market for KNX Members. The current portfolio of twisted-pair-based products often necessitates major renovations, limiting their application in retrofitting scenarios.

By leveraging Thread, KNX IoT devices provide a wireless, scalable solution that reduces installation barriers and expands the potential market. This is particularly critical as global trends emphasize sustainability, energy efficiency, and smart infrastructure development. The integration of KNX IoT with Thread makes it possible for KNX Members to remain competitive in the evolving IoT landscape while meeting the demands of a connected world

2 Technology Overview

The KNX IoT framework represents the evolution of the KNX standard, extends KNX over IPv6, and supports multiple subnetworks (Thread, Wi-Fi, Ethernet/SPE). At the heart of this innovation is the integration of Thread.

2.1 Thread Technology Overview

Thread provides a robust foundation for KNX IoT devices, supporting secure, reliable, and scalable device-to-device communication. Unlike proprietary wireless protocols, Thread is built on open standards and designed specifically for low-power, low-latency applications, making it an ideal choice for smart homes and commercial buildings. It ensures devices can communicate efficiently using IPv6, enabling seamless integration with broader IoT ecosystems.

Key aspects of Thread include:

- IPv6 backbone: Thread operates natively on IPv6, ensuring global scalability and addressing capabilities essential for modern IoT applications.
- Mesh networking: Thread's self-healing mesh topology ensures reliable communication, even when individual devices or connections fail. This eliminates single points of failure and enhances overall network stability.
- Low power: Thread devices are optimized for battery efficiency, enabling years of operation on standard batteries, crucial for retrofitting existing installations.
- Seamless integration: Thread networks leverage border routers to connect with external IP networks, allowing KNX IoT devices to coexist with conventional KNX systems.

Introducing Thread Tools

In 2026, Thread Group announced the beta availability of Thread Tools, an app for capturing and sharing real-time, real-world diagnostic data from deployed networks. Thread Tools makes it possible for OEMs, integrators, and advanced

end users to have meaningful visibility into network performance and access actionable insights that take the guesswork out of troubleshooting.

2.2 KNX IoT Integration

KNX IoT extends the traditional KNX ecosystem by enabling devices to operate over Thread and other IPv6-based layers. The use of ETS ensures KNX IoT devices remain fully compatible with the KNX ecosystem. This enables manufacturers to leverage existing tools and expertise while adopting the benefits of IPv6 and mesh networking.

2.3 Thread Features

- **Secure commissioning:** Thread ensures devices are securely authenticated and authorized during commissioning, with all communication encrypted to maintain data integrity.
- **Practical scalability:** capacity per Thread mesh depends on topology, RF conditions, and router/child allocations. For larger sites, use multiple meshes bridged over the IP backbone (Thread 1.4 Backbone/Over-Infrastructure).
- **Global interoperability:** By operating on the IPv6 standard, Thread networks are compatible with other IP-based technologies, facilitating integration with cloud services and other IoT platforms.
- **Thread 1.4 enhancements:** The latest Thread 1.4 specification introduces features including Thread-over-Infrastructure, enhanced diagnostics, and secure commissioning over TLS, further improving its suitability for commercial and industrial applications.

2.4 KNX IoT Advantages

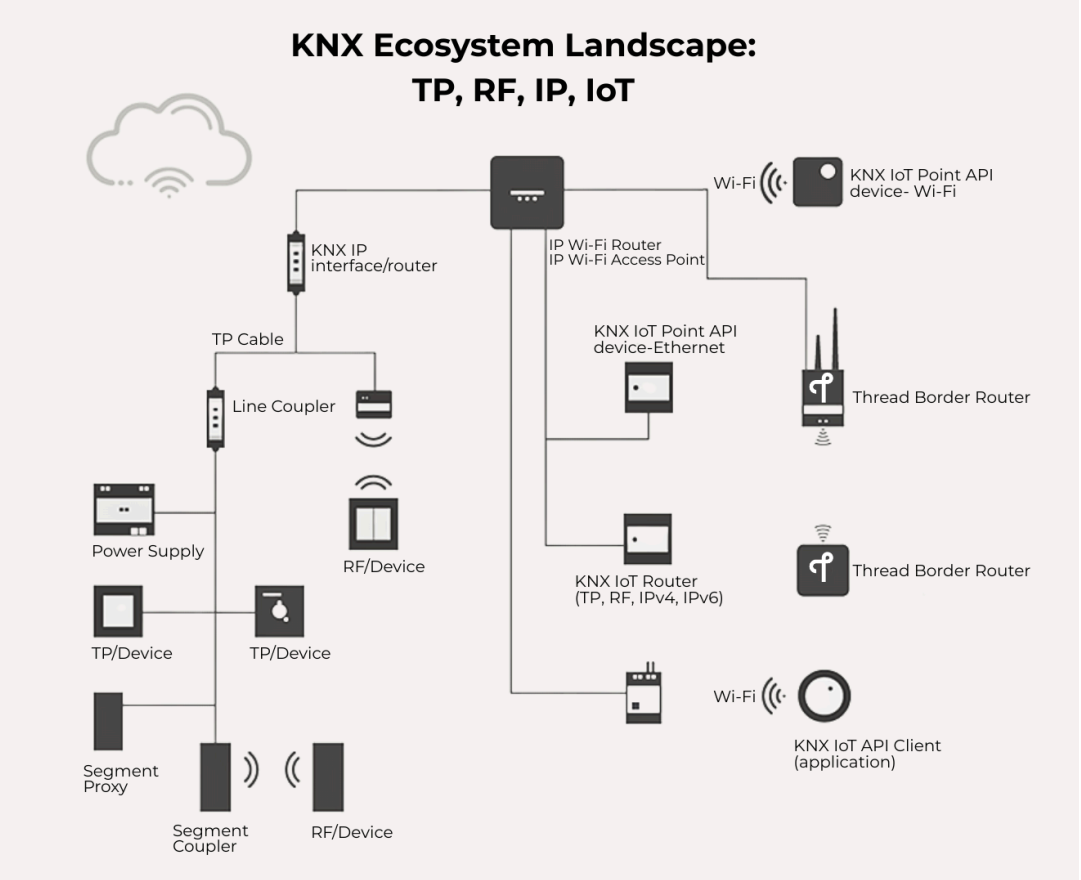
- **Application-level interoperability and ETS workflows** (consistent with conventional KNX).
- **Sustainability and retrofit:** reuse existing backbones and reduce new cabling; enable battery endpoints.
- **Cost optimization:** common tooling and engineering approach across KNX, including KNX IoT.
- **Data-centric services:** standardized access via KIM and KNX IoT API servers (on-prem/cloud).
- **Vendor independence:** a competitive, multi-vendor market.

2.5 Benefits for Stakeholders

- **For manufacturers:** Easy extension of existing product portfolios to include wireless KNX IoT solutions, leveraging Thread's robust, scalable architecture and open-source ecosystem.
- **For installers:** Simplified installation and configuration processes, less training because installers are familiar with the workflow and tools, and training can be done at existing KNX Certified Training Centers.
- **For end users:** Robust KNX-level reliability; support for energy-efficiency strategies via granular control and monitoring; and future-ready IPv6 connectivity.

The alignment of Thread with KNX IoT's principles enables the creation of a scalable, secure, and interoperable ecosystem for modern IoT applications, paving the way for innovative solutions in smart home and building automation.

3 Solution Description



KNX IP Interface/Router Connects KNX networks over	**IP Wi-Fi Router** Routes wireless IP traffic	**KNX IoT point API Device - Wi-Fi**	**TP Cable** Twisted-pair cable for

IP infrastructure	between networks **IP Wi-Fi Access Point** Provides wireless network access to devices	Wi-Fi-connected KNX IoT endpoint device	KNX communication
Line Coupler Connects and filters KNX line traffic	**Media Coupler TP/RF** Bridges twisted-pair and RF KNX networks	**KNX IoT point API Device - Ethernet** Ethernet-based KNX IoT endpoint device	**Thread Border Router** Connects Thread mesh to IP networks
Power Supply Provides regulated power for KNX installations		**RF/Device** Wireless KNX device using radio frequency	**KNX IoT Router (TP, RF, IPv4 y IPv6)** Routes KNX IoT traffic across media
Thread KNX IoT Point API Device Thread-connected KNX IoT endpoint device	**TP/Device** Twisted-pair connected KNX field device	**KNX IoT API Server** Hosts and exposes KNX IoT services	**KNX IoT API Client (application)** Application communicating with KNX IoT services
TP/Devices Multiple twisted-pair connected KNX devices	**Segment Proxy** Represents a remote KNX segment to the network	**Segment Coupler** Connects and manages KNX network segments	**RF/Devices** Multiple wireless KNX radio-frequency devices

3.1 How the Technology Works

- **KNX IoT ecosystem**: Devices implement the Point API (CoAP/CBOR) with OSCORE for end-to-end application security (this is KNX IoT; Thread does not use OSCORE). Devices are ETS-configurable and described via KIM for semantics.
- **Thread network**: Provides the wireless IPv6 mesh (secure at link/network layers).
- **Thread border router:** This device connects the Thread mesh network with external IP networks, such as Ethernet or Wi-Fi, enabling communication between KNX IoT and non-KNX IoT devices. Configuration and commissioning of the Thread Border Router ensure its seamless integration into the KNX IoT ecosystem.
- **KNX IoT router (when needed)**: Bridges KNX IoT and non-IoT KNX networks so both can coexist in one installation.

3.2 Typical Use Cases

Indoor Air Quality (IAQ):
Battery-powered KNX IoT room sensors over Thread for IAQ and HVAC control.

Retrofitting LED Lighting:
Thread's wireless capability allows retrofitting of LED lighting systems in existing buildings without the need for new wiring. These systems can be integrated with KNX IoT to enable energy-efficient, centrally controlled lighting solutions.

Sensor Integration:
Occupancy, contact, metering, security, using Thread modules and KNX IoT Point API for quick portfolio extensions.

Individual Room Temperature Control:
KNX IoT over Thread enables wireless integration of room-temperature sensors, room controllers, valve actuators, and fan-coil controllers. This supports room-by-room heating and cooling

3.3 Technical Specifications

KNX Specifications: 3/10/1 to 3/10/6. All possible downloads are found here, with official KNX IoT developer documentation found here. *Registration required*

Subnetwork choices under IPv6: Thread (wireless), Wi-Fi, Ethernet, SPE/10BASE-T1L.
Security: KNX IoT uses OSCORE over CoAP end-to-end; Thread supplies link/network security.

3.4 Application Examples

Communication Patterns to Use:

The KNX IoT communication supports several messaging patterns that are configured with ETS, each suited to different application and integration needs. **For every KNX Group Address, ETS allows the communication pattern to be configured individually**, enabling the messaging behaviour to be optimized according to the specific application requirements. Choosing the right pattern is a trade-off between confirmation behaviour, message size, and the number of devices addressed. To simplify integration, the **manufacturer can predefine a default setting in the product data**, so that the integrator does not have to configure this manually and can rely on the recommended behaviour out of the box. The following patterns are recommended, with typical usage per domain:

- **Direct unicast** for targeted, confirmed 1:1 exchanges (e.g., fan-speed commands and status). The sender resolves and stores the peer's IPv6 address alongside the Individual Address. Unicast is typically used in **HVAC applications**, where interactions are point-to-point between a room unit and a room controller. In a Thread network, unicast offers two key advantages: it is already **confirmed at the data-link layer and application layer** which makes communication more reliable, and its messages are **shorter**, reducing airtime and energy consumption.

- **Multicast** for N:N updates (e.g., shared status) configured with ETS. Multicast is typically used in **lighting applications**, where a single command addresses many devices simultaneously. The IPv6 multicast Group ID is also used to **optimize the wake-up cycle of sleepy devices**, since the Thread Router only caches messages that have a **registered multicast receiver**. Note, however, that in a Thread network multicast messages are **delivered confirmed only on the first hop**; between mesh extenders (routers), delivery is **not confirmed**.

- **Endpoint subscription (observe)** for controller-centric star patterns when that simplifies integration. Subscription-based communication is well suited to **diagnostic use cases**, for example, when ETS wants to **temporarily listen** to the messages a device sends.

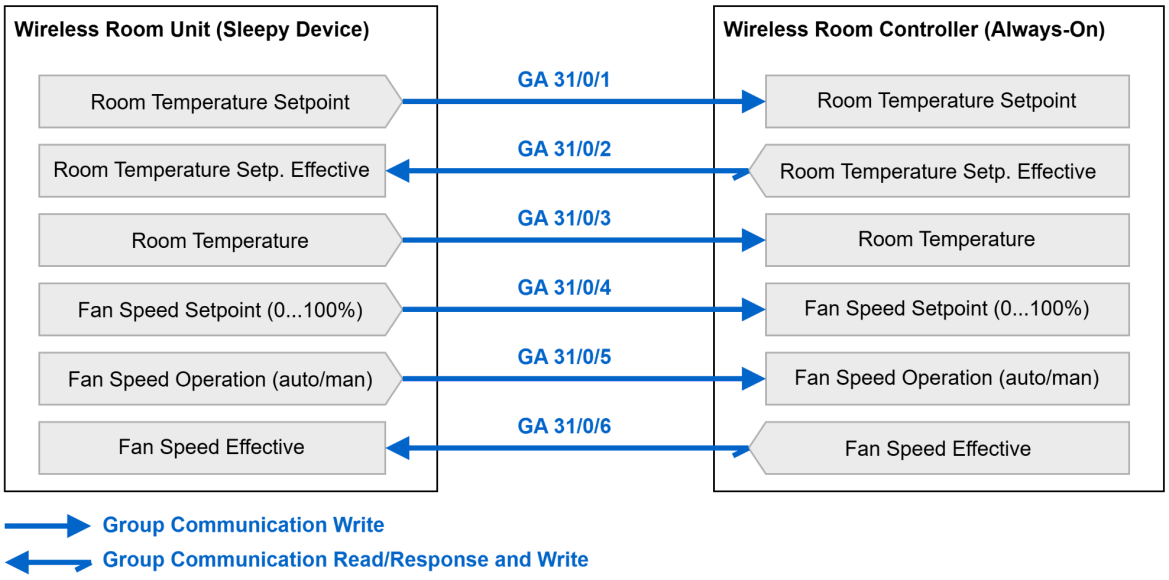
Direct Unicast Communication between Room Unit and Controller:

The example below shows a **Thread sleepy room unit** (battery) interacting with an **always-on fan-coil controller** over KNX IoT. It demonstrates the direct unicast interworking pattern, and optimizations for sleepy devices.

Runtime interworking (what is exchanged): setpoints, commands (fan speed, for example), and status/feedback. To preserve battery life, only necessary values are exchanged, using event-driven updates and selective **periodic reads** from the room unit for display-relevant datapoints.

The figure illustrates the communication architecture between a **battery-powered Wireless Room Unit** and an **always-on Wireless Room Controller**, connected over a **Thread** network. The two devices exchange operational data via group communication (read, response, and write via CoAP POST), using a shared set of datapoints such as Room Temperature Setpoint, Room Temperature (Effective), Fan Speed Setpoint (0…100 %), and Fan Speed Operation (auto/manual), each mapped to a dedicated KNX Group Address (e.g., GA 31/0/1 … GA 31/0/6) configured with ETS.

The **Wireless Room Unit** operates as a **sleepy device** with a nominal sleep cycle of five minutes. To conserve energy, its radio remains inactive for the majority of the time and wakes-up only periodically or on demand. It provides the local user interface, allowing occupants to adjust the setpoint and fan speed and to view current status information on its display.



Because the Room Unit is a sleepy device, it is **recommended that the Room Unit initiates communication**. This approach avoids the latency and buffering constraints associated with reaching a device that is asleep for most of its duty cycle.

- **Room Unit as initiator (recommended):** When a user interacts with the device, for example, by pressing a button the Room Unit wakes-up and initiates the exchange. In doing so, it controls its own wake cycles, writes the updated values (e.g., new setpoint or fan speed) to the Room Controller (group communication write via CoAP POST), and reads back (group communication read/response also with CoAP POST) the latest status from the controller to refresh its local display with current information.

- **Room Controller as initiator (with caching):** Communication in the reverse direction is also possible. However, since the Room Unit may be asleep when the controller sends a message, the **Thread Router must cache the message** and forward it once the Room Unit next wakes-up. This introduces additional latency and buffering, which is why Room Unit-initiated communication is preferred for time-sensitive interactions.

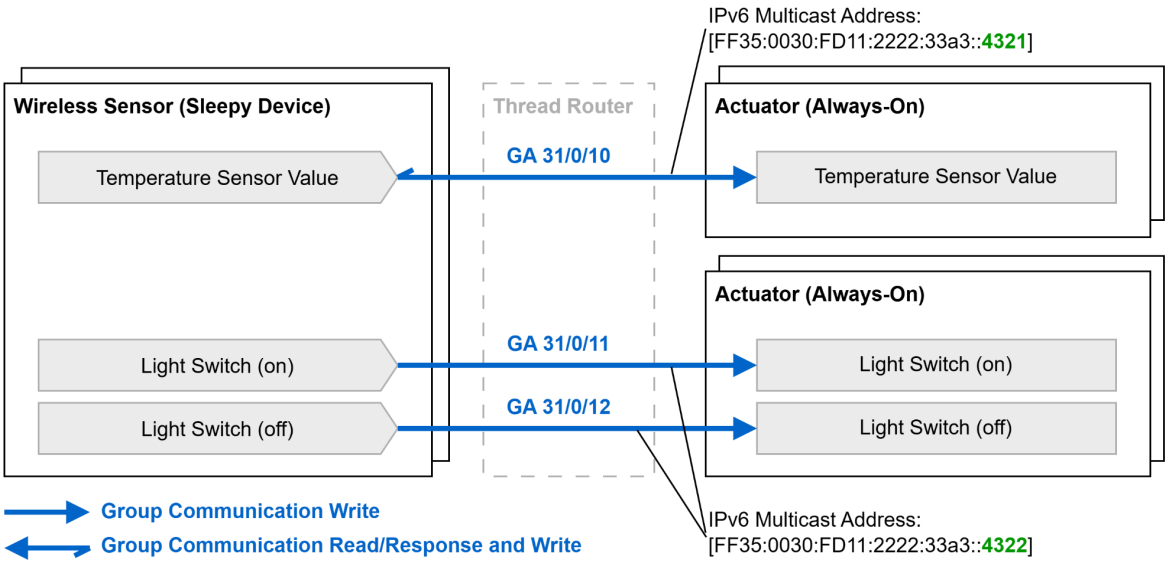
Multicast communication between a Sensor and multiple Actuators Unit:

The example below shows a **Thread sleepy wireless sensor** (battery) interacting with an **always-on actuator** over KNX IoT using **IPv6 multicast**. It demonstrates the multicast interworking patterns and the network-level optimizations that preserve battery life on the sleepy device.

Runtime interworking (what is exchanged): sensor values (temperature) and command datapoints (light switch on/off) are mapped to a dedicated KNX Group Address (e.g., GA 31/0/10 for Temperature Sensor Value, GA 31/0/11 for Light Switch on, and GA 31/0/12 for Light Switch off) configured with ETS. At the network layer, each

KNX Group Address is associated with an **IPv6 multicast address** (e.g., [FF35:0030:FD11:2222:33a3::**4321**] and [FF35:0030:FD11:2222:33a3::**4322**]).

The **Wireless Sensor** is a **sleepy device** whose radio wakes only periodically or on demand to publish datapoints; the **Actuator** is **always powered on** and reacts to the received values.



The figure illustrates the communication architecture between a **battery-powered Wireless Sensors** and an **always-on Actuators**, connected over a **Thread** network via a **Thread Router**. The sensor sends its Temperature Sensor Value and Light Switch events as group communication write (CoAP POST), while the actuator can issue group communication read/response (CoAP POST) on the 31/0/10 KNX Group Address.

Multicast registration as a traffic and battery optimization: The **IPv6 Multicast Group ID is used at the Thread layer to coordinate network traffic**. The Thread Router only forwards multicast traffic for addresses that a device has actively registered, which decides what actually reaches the sleepy sensor.

- **Registered (Temperature Sensor Value, GA 31/0/10):** The sensor registers this IPv6 multicast address so the actuator can send read/response messages, for example on restart, to get a fast initial update. The router therefore forwards these messages to the sensor.

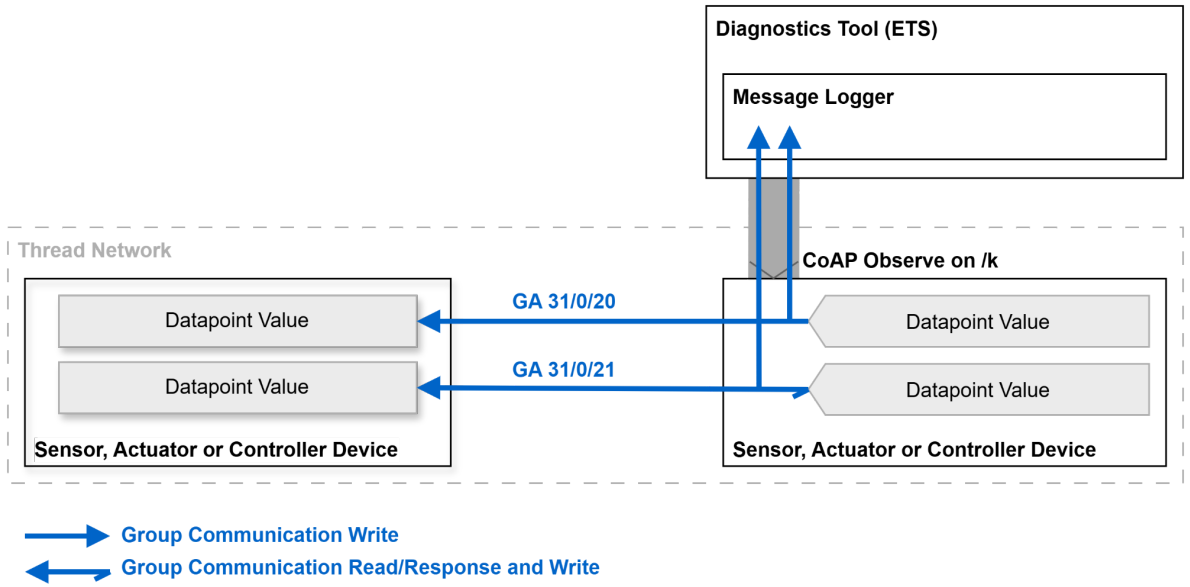
- **Not registered (Light Switch, GA 31/0/11–12):** These KNX Group Addresses or rather the IPv6 multicast address are not registered by the sensor, so multicast messages on them do not reach it. As the sensor only sends switch events, the router has nothing to deliver which is **optimizing network traffic and battery consumption**.

Communication Diagnostics with Endpoint subscription:

The example below shows how a **Diagnostics Tool (ETS)** gains visibility into the runtime communication of a **Sensor, Actuator, or Controller device** over KNX IoT, using the endpoint subscription (CoAP observe) pattern. When devices communicate **inside a subsystem such as Thread**, by **unicast** or **multicast, the Diagnostics Tool cannot passively listen to this traffic**. Yet the tool is **essential for the system integrator** to validate correct behaviour and to find issues or misconfigurations. Endpoint subscription closes this gap by letting the tool actively subscribe to a device instead of overhearing the network.

Single subscription for all runtime data: By design, a **single subscription per device** delivers **all configured runtime communication data the subscriber has access rights to**. Since only one subscription is needed, not one per datapoint, the process is **simple and efficient**, with no **inconsistency** from datapoints left unsubscribed. Different clients can be configured with different access rights and therefore receive only the KNX Group Addresses configured for them. This is how the **communication flow is reduced to exactly what a particular client needs**. This ensures a **complete and consistent view** within the granted access rights, making endpoint subscription the **recommended approach for diagnostics or controller-to-peripheral communication**.

Runtime interworking (what is exchanged): the normal runtime datapoints mapped to their KNX Group Addresses (e.g., GA 31/0/20 and GA 31/0/21) configured with ETS. In addition to sending these to its peer devices, the device **also delivers a copy to the subscribed Diagnostics Tool**, where the messages are logged.



The figure shows a **Sensor, Actuator, or Controller device** in a **Thread** network and a **Diagnostics Tool (ETS)** with a **Message Logger**. The device continues its regular **group communication (write and read/response)** with its peer, while the Diagnostics Tool issues a **CoAP Observe** on the device's **/k resources** to receive every message the device sends, provided it holds the corresponding **access rights**.

- **Normal runtime communication:** The device sends its datapoints to the peer via group communication, staying within the Thread subsystem and **not visible** to external listeners.
- **Observed communication (diagnostics):** The Diagnostics Tool subscribes to the **/k resources via CoAP Observe** and, subject to access rights, receives a copy of all sent messages. The device thus transmits to both its peer and the tool, which logs the messages for validation and fault-finding.

4 Case Study

The following example illustrates one path; other vendors provide comparable system components and services.

4.1 Atouch Winwell Journey

Atouch Winwell illustrates a practical and successful implementation of Thread within the KNX ecosystem. Atouch.Winwell partnered with Cascoda to develop a groundbreaking KNX IoT product using the Chili2S Thread-based module. The solution addressed challenges such as:

- Remodeling buildings without rewiring.
- Overcoming range limitations without repeaters.
- Meeting enhanced security standards.

Atouch.Winwell began its journey at Light + Building 2024, where the company leveraged pre-certified hardware and development services from Cascoda to expedite its development process.

4.2 Challenges

1. **Thread Radio + Open-Source Stack Integration:**
   Implementers who choose to develop their own solution can integrate the open-source KNX stack with their selected Thread radio, providing flexibility in hardware and system design. This requires integration and validation work to ensure compatibility between the software stack and the chosen platform, but considerably reduces the effort compared with developing a KNX stack from scratch. Alternatively, certified off-the-shelf system components, such as Cascoda's solutions, provide a pre-integrated option that can help streamline development and accelerate deployment.

2. **Certification Requirements:**
   The combined solution (Thread radio + open-source stack) must undergo the applicable KNX certification process to demonstrate compliance with

KNX requirements. As with any product development, the effort involved will depend on the selected hardware platform, the level of prior integration and the implementer's experience. Pre-certified components and experienced technical support can help streamline this process and shorten the path to certification.

3. **System Familiarity:**
   Many KNX community members are new to Thread-based integration, making onboarding and configuring components like the Thread border router challenging.

How to Avoid These Challenges

1. **Leverage pre-certified solutions:**
   Use pre-certified Thread modules to reduce development and certification time, enabling faster market entry.
2. **Partner with experts:**
   Collaborate with organizations offering pre-integrated solutions, technical support, and tools. For instance, Cascoda provides pre-certified Thread modules and development services that simplify the process.
3. **Focus on training:**
   Developers (manufacturers' firmware/embedded teams): training on KNX IoT Point API, KIM modeling, CoAP/CBOR, OSCORE usage, Thread commissioning, diagnostics, and Border Router operation.

   KNX System Integrators (i.e., the professionals who configure KNX projects in ETS): training on ETS6 workflows for KNX IoT devices, project structuring that spans KNX and KNX IoT, and Thread onboarding basics (what information they need from the border router owner, how discovery works, troubleshooting steps).
4. **Iterative Testing:**
   Perform incremental testing during the development phase to identify and address compatibility issues early, reducing the risk of delays during certification.

- **Module/board bring-up:** sanity checks for radio/MCU, basic CoAP endpoints, power profiles (sleep/wake), and timing.
- **Protocol conformance (developer level):** verify KNX IoT Point API resources/behaviors, CoAP + CBOR encoding, and OSCORE end-to-end protection; run stack self-tests where provided.
- **EITT-based testing:** Use **EITT** (KNX Interworking Test Tool) where applicable to script functional/interoperability test cases for KNX IoT behavior and regression runs. Combine with your vendor's Thread test harness for mesh behavior.
- **ETS6 system tests:** build a small ETS project with representative devices; validate group communication, parameterization, scenes/alarms, and semantic export. Include **negative tests** (wrong credentials, missing resources, power-loss recovery).

- **RF/mesh validation:** in a representative floor plan, test multiple topologies (router density, distance, wall materials); monitor latency, packet loss, and re-route times; tune router/child allocations; capture traffic with a Thread protocol analyzer; verify Border Router failover where relevant.
- **Pre-certification checklist:** confirm KNX device profile coverage, security posture (key storage, firmware update flow), ETS artifacts, and documentation. If your module is not pre-certified for Thread, execute the vendor's Thread certification test plan.

4.3 Results and Outcomes

Measurable Benefits:

- **Market reach:** Atouch.Winwell tapped into the €55 million KNX IoT keypad market, unlocking new business opportunities.
- **Speed to market:** They achieved a working prototype within two months and completed certification in seven months.
- **Brand awareness:** Collaboration with KNX and Cascoda enhanced Atouch.Winwell's international recognition.
- **Development efficiency:** Faster development and certification by reusing certified system components and a KNX IoT stack.
- **Retrofit readiness:** Retrofit-friendly wireless deployments (no rewiring) improved competitiveness in renovation bids.

"A pillar for fast and efficient development on complex ecosystems," - Duarte Geraldes, General Manager at Atouch Winwell.

5. Comparison and Competitive Analysis

KNX IoT operates over IPv6 and supports multiple subnetworks; selection is made per device role and project constraints, while ETS workflows and KNX application-level interoperability remain consistent across them.

5.1 How KNX IoT uses each subnetwork

Thread (wireless IPv6 mesh)

- **Best for:** Battery and low-power endpoints; dense sensing/actuating; retrofit scenarios.
- **Why under KNX IoT:** Self-healing mesh, site-backbone scaling via Border/Backbone Routers.
- **Considerations:** Commissioning/onboarding learning curve; plan RF topology and router/child allocations.

Wi-Fi (wireless IPv6 star)

- **Best for:** High-throughput or UI-rich devices (panels, gateways), typically mains-powered.
- **Why under KNX IoT:** Ubiquitous infrastructure and straightforward backbone integration.
- **Considerations:** Power use and shared-medium contention in dense deployments.

Ethernet (multi-pair, switched)

- **Best for:** Panels, edge servers, and backbones that require deterministic, low-latency links; PoE simplifies power + data.
- **Why under KNX IoT:** Mature tooling/monitoring; rock-solid reliability in control rooms and cabinets.
- **Considerations:** Cabling effort/cost—most attractive in new build or major refurb.

Single-Pair Ethernet / 10BASE-T1L (SPE)

- **Best for:** Long-reach wired (up to ~1 km), constrained pulls, industrial/heritage sites; PoDL for power + data over one pair.
- **Why under KNX IoT:** Brings IPv6 to places where multi-pair Ethernet is impractical and wireless is constrained.
- **Considerations:** Lower throughput than multi-pair; ecosystem still maturing vs. classic Ethernet/Wi-Fi.

5.2 Bottom line within KNX IoT

All four subnetworks serve KNX IoT over IPv6. Choose per device role and project constraints:

- Retrofit + battery sensors → Thread
- High-bandwidth/mains devices → Wi-Fi
- Panels/backbone/strict latency → Ethernet
- Long-reach wired with simpler pulls → SPE (10BASE-T1L)

KNX keeps your application interoperability, ETS workflows, and certification consistent across these choices, so portfolios can span multiple subnetworks without fragmenting the user or integrator experience.

6 Conclusion

This white paper explained how KNX IoT extends the core strengths of KNX—including application-level interoperability, ETS-based workflows, and vendor independence—into the IPv6 world. It demonstrated why Thread is a

strong choice for wireless field-level networking, retrofits, and battery-powered endpoints, and how KNX IoT can seamlessly coexist with existing KNX installations through a KNX IoT Router within a single ETS project. The paper also highlighted how available system components, tools, and training resources help accelerate time-to-market. Finally, it positioned Wi-Fi, Ethernet, and SPE as complementary network infrastructures within the broader KNX IoT ecosystem.

KNX IoT over Thread should be viewed as an extension of the KNX toolkit, not as a replacement for every existing medium. Its value lies in combining Thread's low-power IPv6 mesh networking with the KNX application model, ETS workflows, and application-level interoperability.

The next step is practical validation through pilot implementations, EITT-based KNX testing, Thread network validation, and documented installer feedback. The resulting findings should inform future versions of this living white paper.

6.2 Recommended Next Steps

1. Start building: Select a pre-certified Thread system component and a KNX IoT stack (open-source or proprietary) and map your product's datapoints to KIM.
2. Plan integration: Define your Thread Border Router setup and, if you need interworking, your KNX IoT Router approach (stand-alone or integrated).
3. Use ETS6 early: Create a pilot ETS project to validate parameters, group objects, and semantics.
4. Leverage the ecosystem:
    a. Check the KNX IoT Roll-Out area for specs, tooling, and off-the-shelf system components.
    b. Engage with KNX Members and Thread suppliers for modules, TBRs, and routers.
    c. Book training for manufacturers' developers and KNX System Integrators (ETS users).
5. Demo & certify: Schedule a demo, run EITT/ETS tests, and submit for KNX device (and, if needed, Thread) certification.

6.3 Collaboration opportunities

KNX IoT thrives on collaboration, and there are numerous opportunities for manufacturers, solution providers, and developers to contribute to and benefit from the ecosystem:

- **Component and tooling partners:** System-component vendors (e.g., Thread modules, KNX IoT Routers, border routers) ready to support integration and certification.
- **Standards ecosystem:** Participate in KNX and Thread Group activities to align roadmaps and accelerate interoperability.

- **Solution delivery:** Work with KNX partners to design pilots (retrofit floors, IAQ programs, lighting efficiency).

7. References and Further Reading

7.1 Citations

The following sources were used in the preparation of this document:

1. KNX IoT Roll-Out 2024 Support Area: https://support.knx.org/hc/en-us/articles/11224992993426-KNX-IoT-Roll-out-2024
2. KNX IoT Innovations: KNX IoT Innovations Document
3. KNX IoT Roll-Out 2023: KNX IoT Roll-Out 2023 Document
4. KNX IoT System and Application Configuration - White paper
5. Thread 1.4 Public Specification: Thread 1.4 Public Specification PDF
6. Thread 1.4 Features White Paper: Thread 1.4 Features White Paper PDF
7. Cascoda Chili2S Product Information: https://www.cascoda.com/products/module/chili2s/
8. Cascoda KNX IoT Hub: https://www.cascoda.com/products/knx-iot-hub/
9. Success Story: Atouch KNX IoT Case Study

7.2 Additional Resources

For further reading and exploration, consider the following resources:

- KNX Website for Manufacturers: https://www.knx.org/knx-en/for-manufacturers/knxiot/
- KNX for Professionals: https://www.knx.org/knx-en/for-professionals/benefits/knx-iot/
- Thread Group: https://www.threadgroup.org

8. Appendices

8.1 Glossary

- KNX IoT: KNX over IPv6; device profiles include KNX IoT Point API device and KNX IoT Router.
- CoAP/CBOR: Lightweight REST protocol and compact binary encoding used by KNX IoT.

- OSCORE: End-to-end application-layer protection for CoAP in KNX IoT.
- Thread: IPv6 wireless mesh (IEEE 802.15.4) with self-healing routing and Border/Backbone Routers.
- Thread Border Router (TBR): Bridges Thread mesh to the site IP backbone (Ethernet/Wi-Fi).
- KNX IoT Router: Bridges KNX in one installation.
- SPE (10BASE-T1L): Single-pair Ethernet with long reach and optional PoDL.
- EITT: KNX Interworking Test Tool used to script functional/interoperability tests for KNX (including KNX IoT) and for regression testing.
- KNX System Integrator: the professional who designs/configures KNX projects in ETS (distinct from device manufacturers' firmware/embedded developers).

8.2 Where to find off-the-shelf components

Consult the KNX IoT Roll-Out support area for manufacturers offering certified Thread system components, Thread Border Routers, and KNX IoT Routers.

9. About the Organizations

About KNX Association: KNX is the globally recognized standard for home and building automation, offering a vendor-independent platform for seamless interoperability. With over 500 member companies and thousands of certified products, KNX has established itself as a leader in smart building solutions. The association continues to evolve, integrating IoT technologies like Thread and Wi-Fi into its ecosystem to address modern demands for scalable, secure, and efficient automation.

Website: https://www.knx.org/

KNX Association

- Email: info@knx.org
- Website: https://www.knx.org/
- Resources: KNX Support Area

About Thread Group: Thread Group is an industry alliance dedicated to developing and promoting Thread, a low-power, IPv6-based wireless mesh networking protocol designed for IoT applications. The Thread Group comprises leading companies in technology and IoT, working collaboratively to ensure Thread's adoption and interoperability across smart devices. Thread is particularly suited for applications requiring reliability, scalability, and energy efficiency.

Thread Group

- Email: support@threadgroup.org
- Website: https://www.threadgroup.org/