SILICON LABS

tech talks

WELCOME

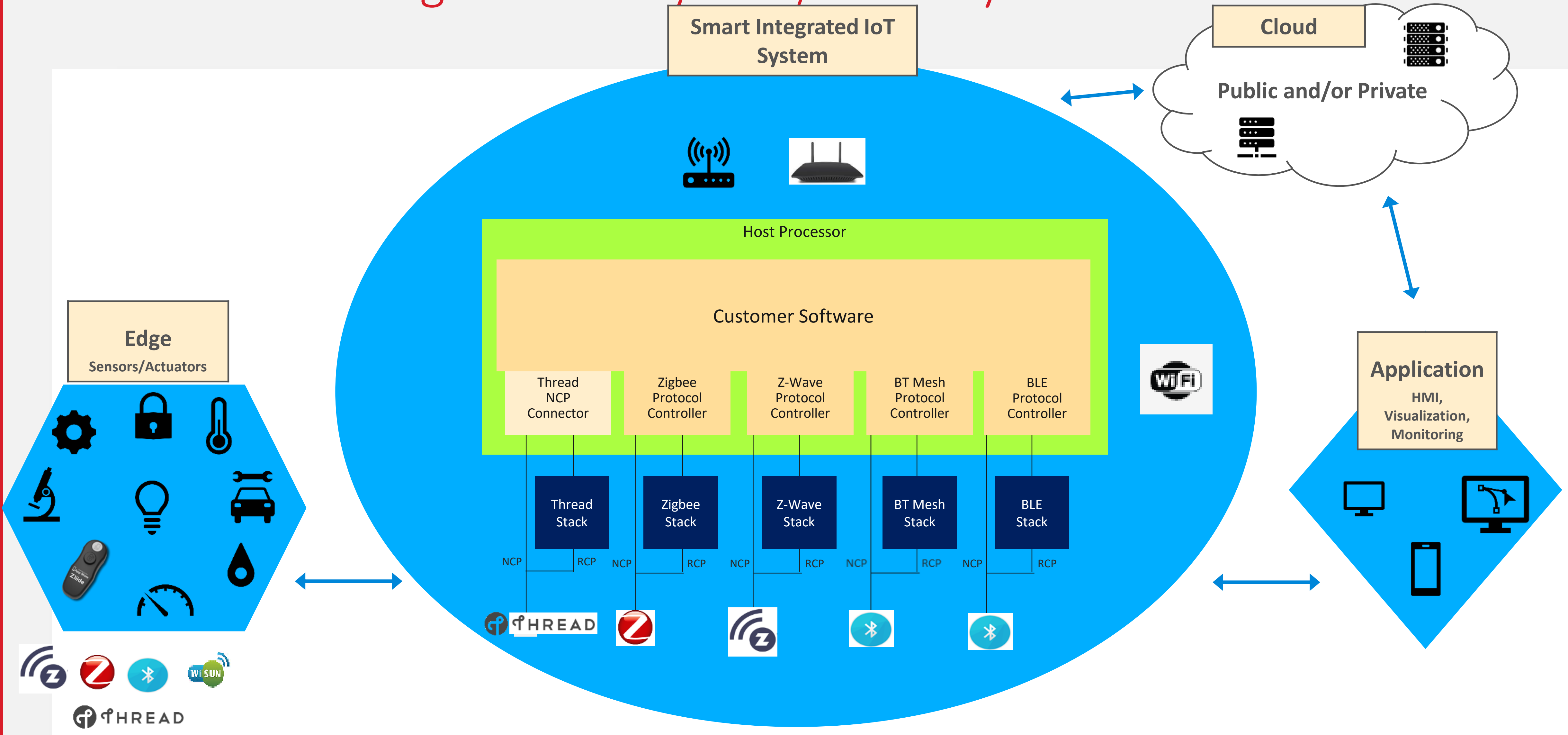
How Unify SDK Helps Manage Multiple Protocols

Wendy Warne

Unify SDK

Presented by: Wendy Warne

Before: Smart Integrated IoT System/Gateway to Cloud



Smart Integrated IoT System

Cloud

Public and/or Private

Host Processor

Customer Software

Thread NCP Connector

Zigbee Protocol Controller

Z-Wave Protocol Controller

BT Mesh Protocol Controller

BLE Protocol Controller

Thread Stack

Zigbee Stack

Z-Wave Stack

BT Mesh Stack

BLE Stack

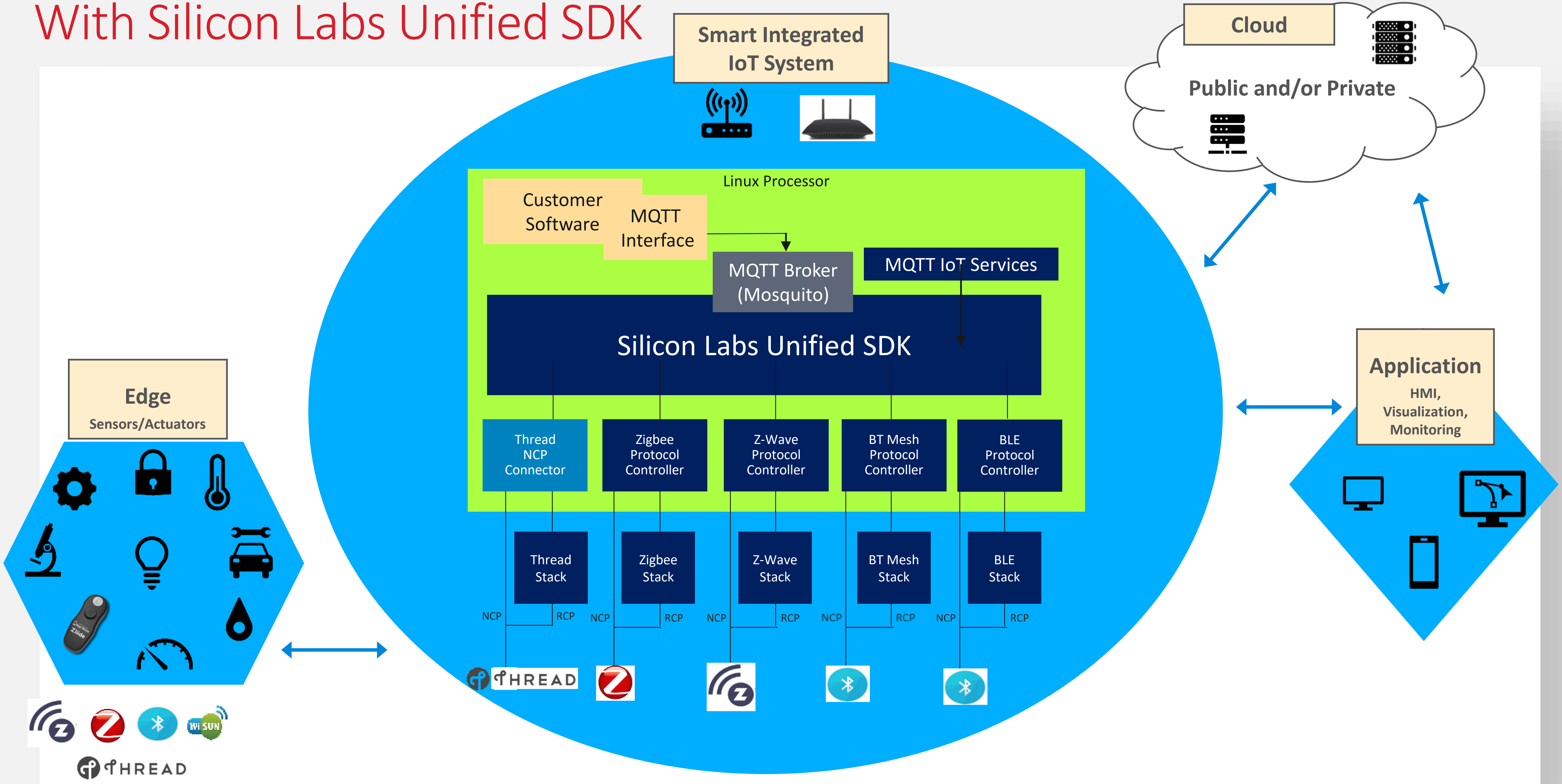
Edge

Sensors/Actuators

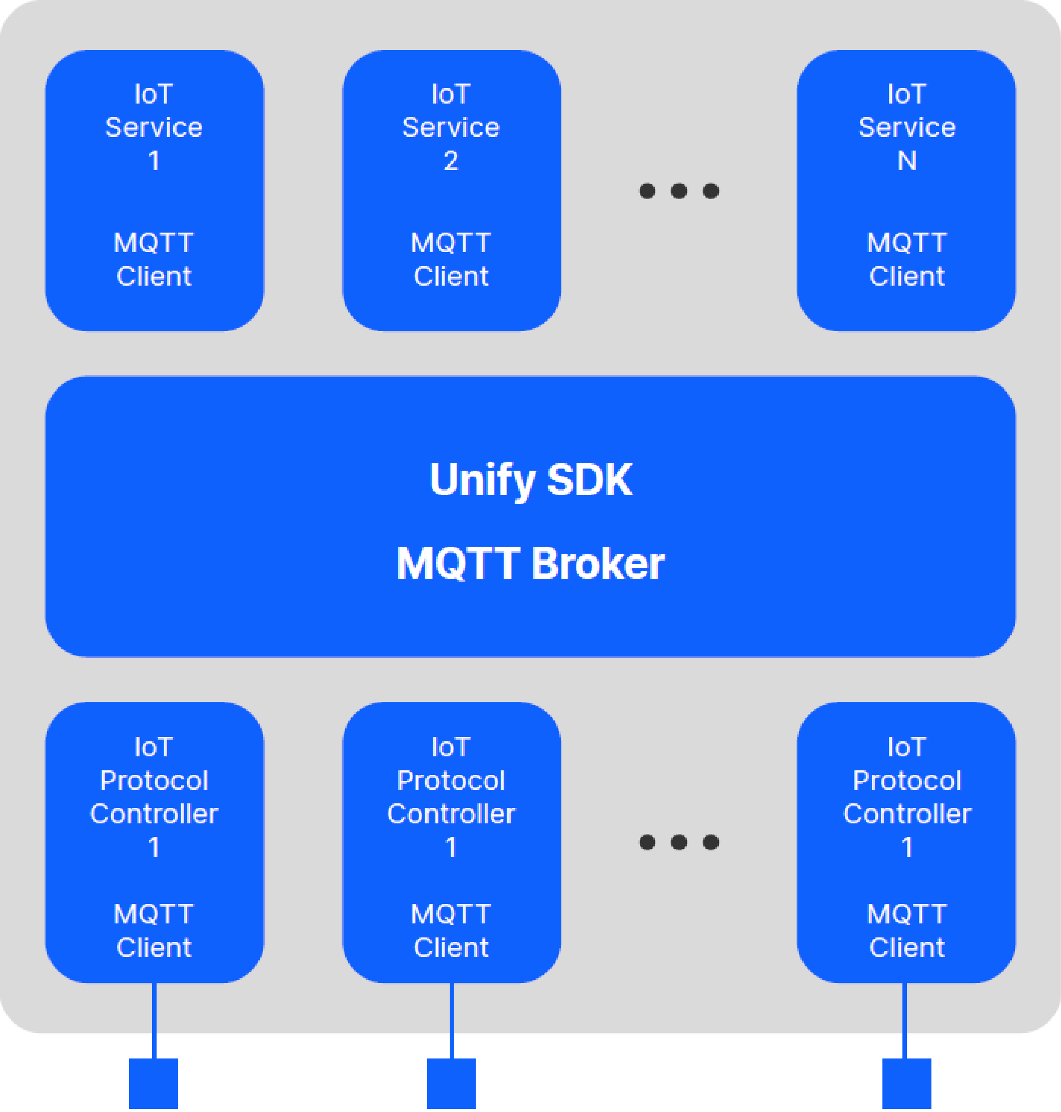
Application

HMI, Visualization, Monitoring

With Silicon Labs Unified SDK



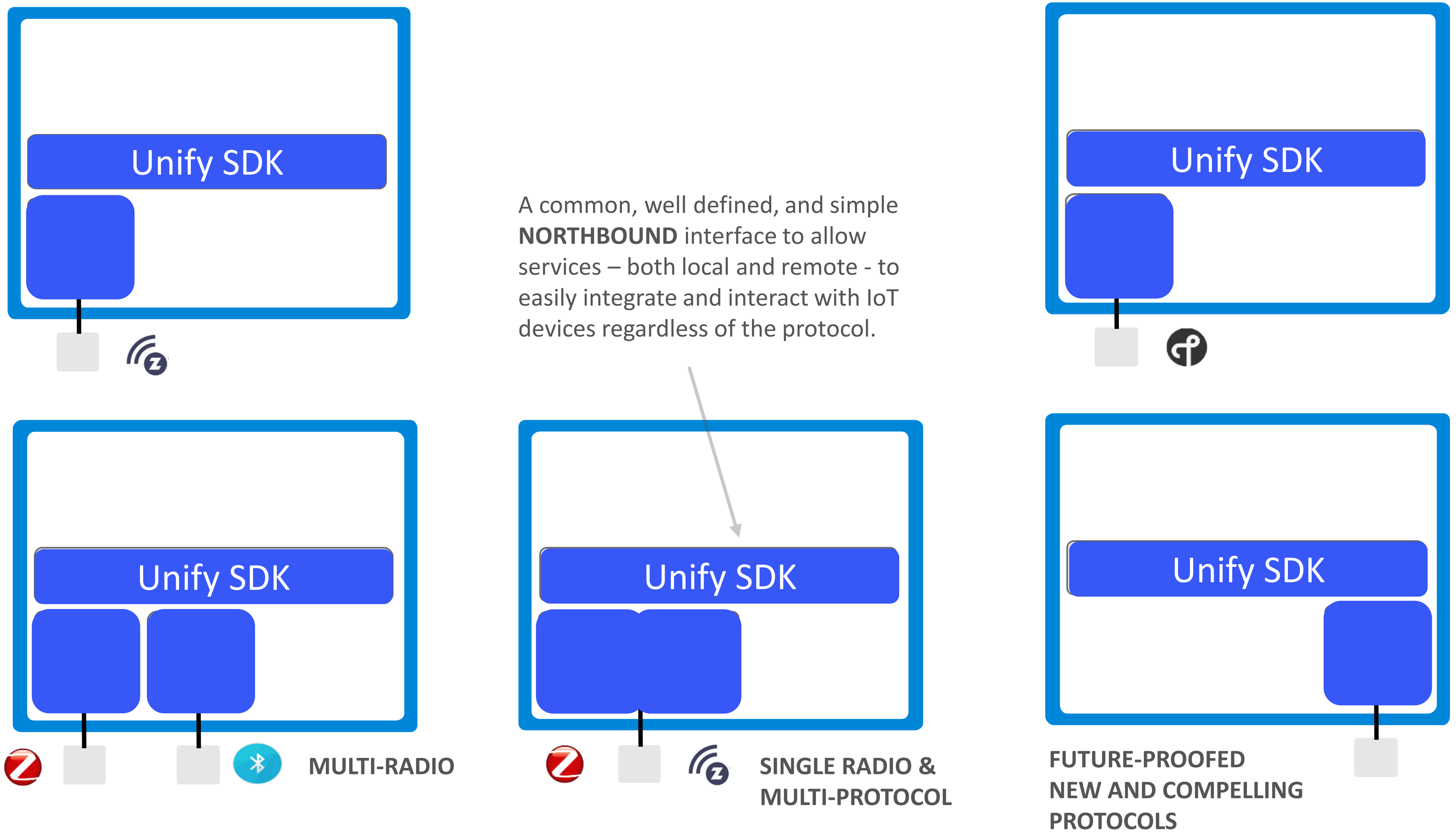
Architecture



Unify SDK for Infrastructure:

- MQTT based UIC Framework
- Common IoT Service Interface for Applications
- Protocol Controllers
  - Abstracts all protocols in stable common API
  - Flexible architecture that can expand to other protocols

Protocol Abstraction with Unify SDK – Future Proofing



A common, well defined, and simple **NORTHBOUND** interface to allow services – both local and remote - to easily integrate and interact with IoT devices regardless of the protocol.

Unify SDK

Unify SDK

Unify SDK

MULTI-RADIO

Unify SDK

SINGLE RADIO & MULTI-PROTOCOL

Unify SDK

FUTURE-PROOFED NEW AND COMPELLING PROTOCOLS

Example: The Z-Wave Protocol Controller

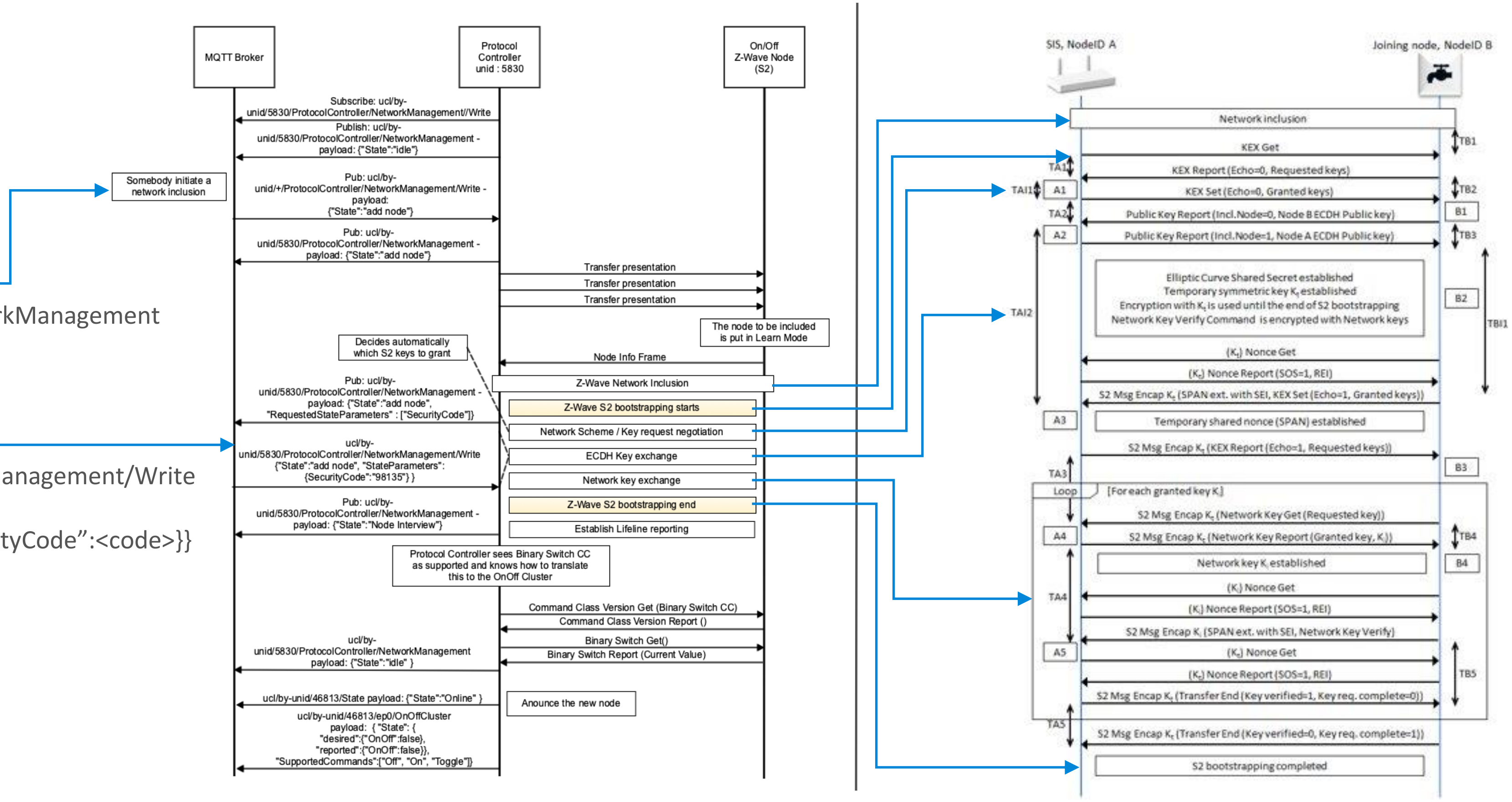
User Input

Publish (PUB):
ucl/by-unid/<UID>/ProtocolController/NetworkManagement
Payload:
{“State”:”add node”}

PUB:
ucl/by-unid/<UID>/ProtocolController/NetworkManagement/Write
Payload:
{“State”:”add node”, “StateParameters”: {“SecurityCode”:<code>}}

Protocol Controller

Z-Wave Frames



MQTT Broker

Protocol Controller unid : 5830

On/Off Z-Wave Node (S2)

Subscribe: ucl/by-unid/5830/ProtocolController/NetworkManagement//Write

Publish: ucl/by-unid/5830/ProtocolController/NetworkManagement - payload: {"State":"idle"}

Somebody initiate a network inclusion

Pub: ucl/by-unid/+/ProtocolController/NetworkManagement/Write - payload: {"State":"add node"}

Pub: ucl/by-unid/5830/ProtocolController/NetworkManagement - payload: {"State":"add node"}

Transfer presentation

Transfer presentation

Transfer presentation

The node to be included is put in Learn Mode

Decides automatically which S2 keys to grant

Node Info Frame

Z-Wave Network Inclusion

Pub: ucl/by-unid/5830/ProtocolController/NetworkManagement - payload: {"State":"add node", "RequestedStateParameters" : ["SecurityCode"]}

Z-Wave S2 bootstrapping starts

Network Scheme / Key request negotiation

ucl/by-unid/5830/ProtocolController/NetworkManagement/Write {"State":"add node", "StateParameters": {SecurityCode":"98135"} }

ECDH Key exchange

Network key exchange

Z-Wave S2 bootstrapping end

Pub: ucl/by-unid/5830/ProtocolController/NetworkManagement - payload: {"State":"Node Interview"}

Establish Lifeline reporting

Protocol Controller sees Binary Switch CC as supported and knows how to translate this to the OnOff Cluster

Command Class Version Get (Binary Switch CC)

Command Class Version Report ()

ucl/by-unid/5830/ProtocolController/NetworkManagement payload: {"State":"idle" }

Binary Switch Get()

Binary Switch Report (Current Value)

ucl/by-unid/46813/State payload: {"State":"Online" }

Anounce the new node

ucl/by-unid/46813/ep0/OnOffCluster payload: { "State": { "desired":{"OnOff":false}, "reported":{"OnOff":false}}, "SupportedCommands":["Off", "On", "Toggle"]}

SIS, NodeID A

Joining node, NodeID B

Network inclusion

TB1

KEX Get

TA1

KEX Report (Echo=0, Requested keys)

TAI1

A1

KEX Set (Echo=0, Granted keys)

TB2

TA2

Public Key Report (Incl.Node=0, Node B ECDH Public key)

B1

A2

Public Key Report (Incl.Node=1, Node A ECDH Public key)

TB3

Elliptic Curve Shared Secret established
Temporary symmetric key K_t established
Encryption with K_t is used until the end of S2 bootstrapping
Network Key Verify Command is encrypted with Network keys

TAI2

B2

TBI1

(K_t) Nonce Get

(K_t) Nonce Report (SOS=1, REI)

S2 Msg Encap K_t (SPAN ext. with SEI, KEX Set (Echo=1, Granted keys))

A3

Temporary shared nonce (SPAN) established

S2 Msg Encap K_t (KEX Report (Echo=1, Requested keys))

TA3

B3

Loop [For each granted key K_i]

S2 Msg Encap K_t (Network Key Get (Requested key))

A4

S2 Msg Encap K_t (Network Key Report (Granted key, K_i))

TB4

Network key K_i established

B4

TA4

(K_i) Nonce Get

(K_i) Nonce Report (SOS=1, REI)

S2 Msg Encap K_i (SPAN ext. with SEI, Network Key Verify)

A5

(K_t) Nonce Get

(K_t) Nonce Report (SOS=1, REI)

TB5

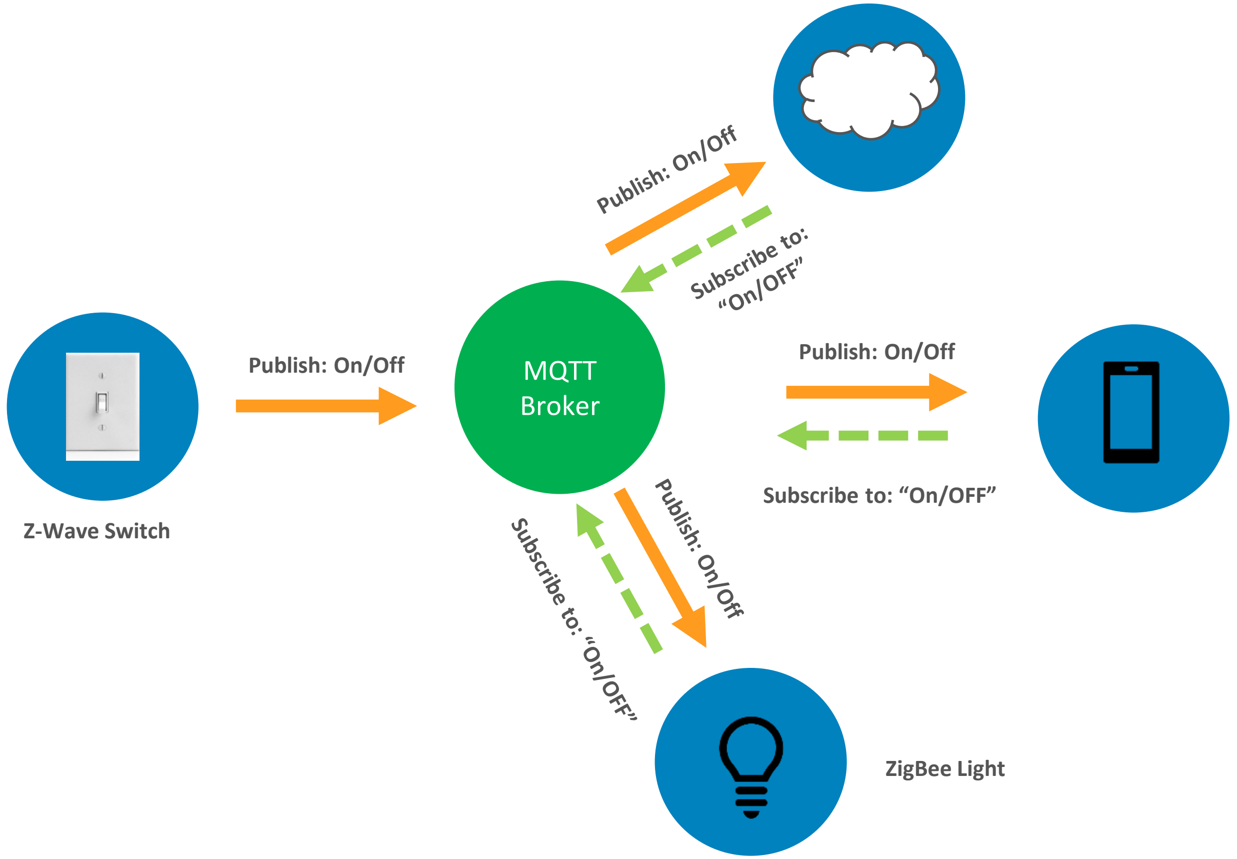
S2 Msg Encap K_t (Transfer End (Key verified=1, Key req. complete=0))

TA5

S2 Msg Encap K_t (Transfer End (Key verified=0, Key req. complete=1))

S2 bootstrapping completed

MQTT Overview

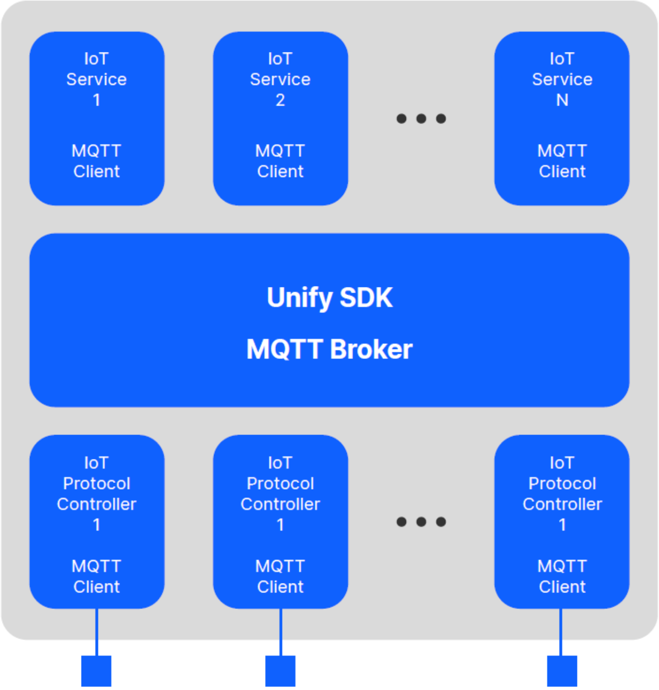


MQTT Syntax and Format

- Format
  - Topic/where/what
- Wild Cards
  - Plus sign (+): It is a single level wildcard
    - Publish two topics **sensors/location1/temp** and **sensors/location2/temp**,
    - Subscribe to **sensors/+/temp**
  - Hash (#) – multi level wildcard
    - Two topics **sensors/location1/temp** and **sensors/location1/humidity**
    - Subscribe **sensors/location1/#**

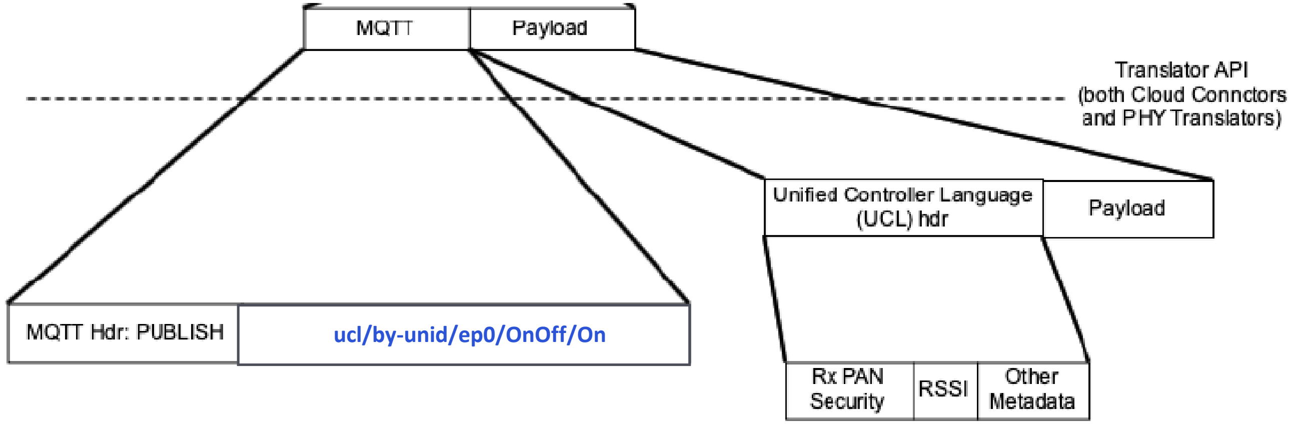
Elements of Unify SDK

- Broker
- The Access Control List (ACL) - MQTT Broker Security
- Resource Directory

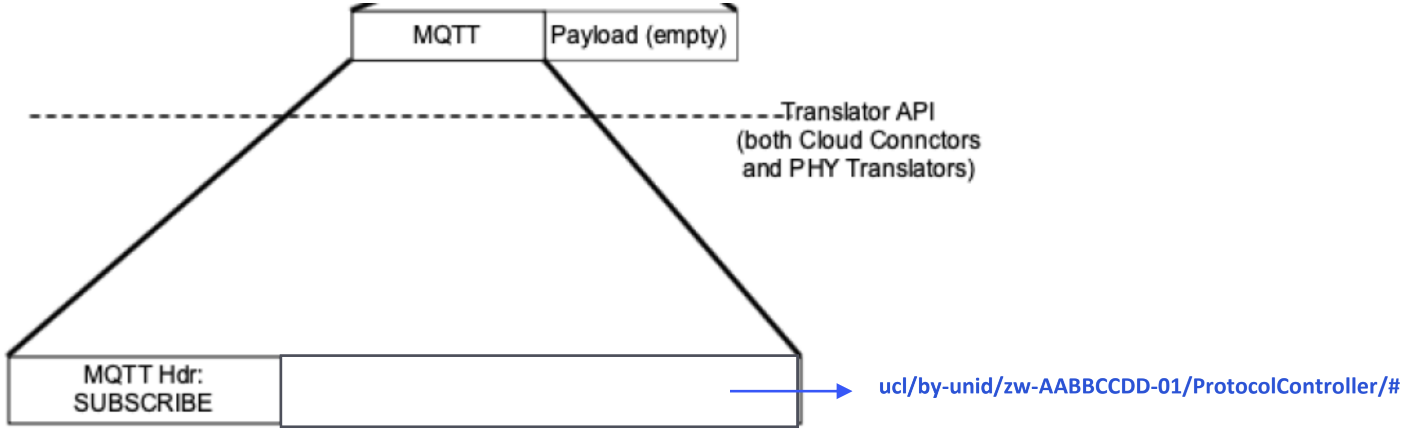

IoT Service 1 MQTT Client
IoT Service 2 MQTT Client
• • •
IoT Service N MQTT Client
Unify SDK
MQTT Broker
IoT Protocol Controller 1 MQTT Client
IoT Protocol Controller 1 MQTT Client
• • •
IoT Protocol Controller 1 MQTT Client

The Unified Controller Language (UCL) - MQTT

- Published messages look as follows:



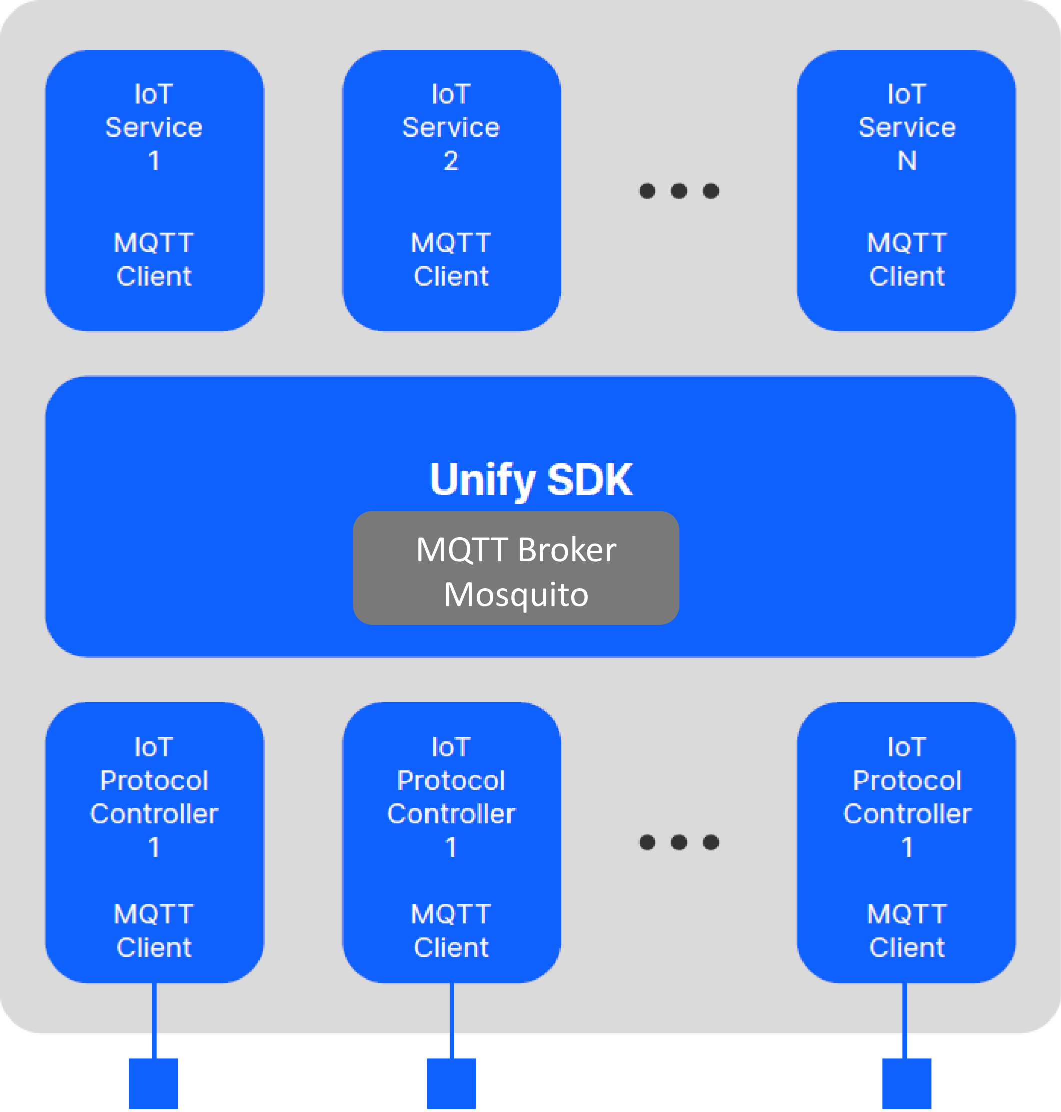
- Subscribe messages look as follows:



UCL Payloads

- UCL Payloads are JavaScript Object Notation (JSON) formatted whose fields follow the Dot-Dot Library specification
- Examples of Dot-Dot functions:
  1. OnOff (0x0006)
  2. On/Off Switch (0x0007)
  3. Level (0x0008)
  4. Door Lock (0x0101)
  5. Thermostat (0x0201)
  6. Occupancy Sensing (0x0406)

Unify SDK Project - Today



- IoT Services
  - dev_ui
  - UPVL – provisioning
  - ANGEL – group manager
  - OTA

- IoT Protocol Controllers
  - Z-Wave
  - ZigBee - Beta

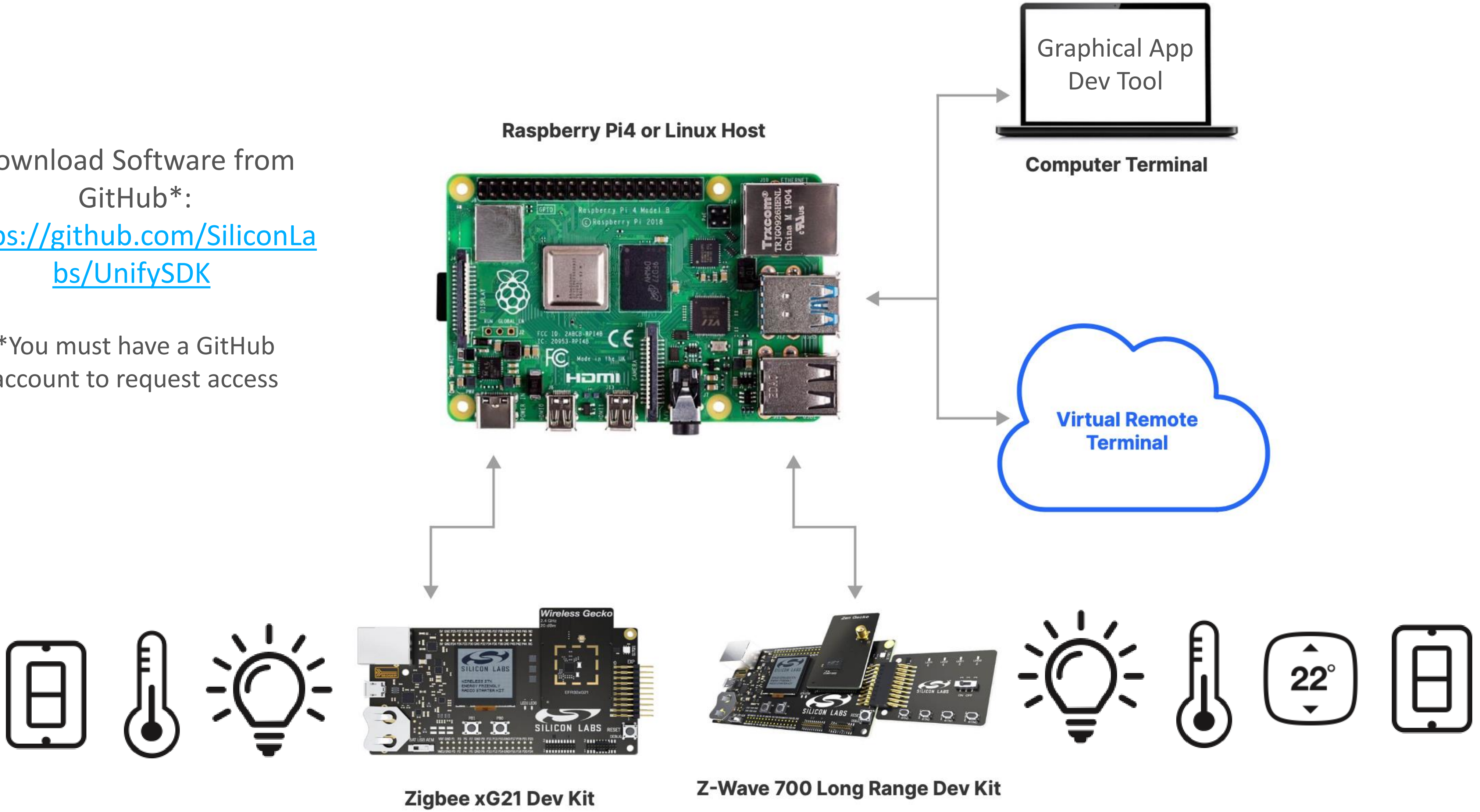
Download Software from GitHub*:
https://github.com/SiliconLabs/UnifySDK

*You must have a GitHub account to request access

Getting Started with Unify SDK for IoT Gateway Development

Download Software from GitHub*: https://github.com/SiliconLabs/UnifySDK

*You must have a GitHub account to request access



Unify SDK Benefits

- Accelerates time-to-market
- Save money in IoT gateway software development
- Streamline maintenance – including software updates and protocol revisions
- Minimize the need for expert wireless development resources
- Scale your IoT wireless software development and maintenance

Join our next Tech Talk

tech talks

WEBINAR

Walk Through Silicon Labs'
New Support for Apple HomeKit

Register Now

SILICON LABS