**Technical & Functional analysis**

**Breakout game with accelero controller using Thread/Matter**

|  |  |
| --- | --- |
|  |  |

Date : 3/14/2024

Contacts : Dekimo : Gysen Bart

Table of Contents

[1. Project overview 4](#_Toc161326106)

[1.1. Matter protocol 4](#_Toc161326107)

[1.2. OpenThread network protocol 4](#_Toc161326108)

[2. Game controller 5](#_Toc161326109)

[2.1. EFR32MG24 - BRD2601B Dev Kit 5](#_Toc161326110)

[2.2. EFR32MG24 Wireless SoC 6](#_Toc161326111)

[2.3. ICM-20689 6-Axis sensor 7](#_Toc161326112)

[2.4. User Button 8](#_Toc161326113)

[3. (Open) Thread 9](#_Toc161326114)

[3.1. OpenThread Border Router 10](#_Toc161326115)

[3.2. Rights – OpenThread – Thread 10](#_Toc161326116)

[4. Raspberry Pi 4 11](#_Toc161326117)

[5. Matter protocol 11](#_Toc161326118)

[5.1. Terms 11](#_Toc161326119)

[5.2. About 12](#_Toc161326120)

[5.3. Documents 12](#_Toc161326121)

[5.4. Architecture 13](#_Toc161326122)

[5.4.1. Network 13](#_Toc161326123)

[5.4.2. Network Topology 14](#_Toc161326124)

[5.5. Data Model 16](#_Toc161326125)

[5.5.1. Node 17](#_Toc161326126)

[5.6. Interaction Model 20](#_Toc161326127)

[5.6.1. Read Interaction 22](#_Toc161326128)

[5.6.2. Subscription Interaction 22](#_Toc161326129)

[5.6.3. Write Interaction 23](#_Toc161326130)

[5.7. Security 25](#_Toc161326131)

[5.7.1. Session establishment 25](#_Toc161326132)

[5.7.2. Message confidentiality and integrity 25](#_Toc161326133)

[5.8. Commissioning 26](#_Toc161326134)

[5.8.1. Prerequisites 26](#_Toc161326135)

[5.8.2. Onboarding payload 26](#_Toc161326136)

[5.8.3. Commissioning steps 27](#_Toc161326137)

[5.9. Matter Hub & Bridge 29](#_Toc161326138)

[5.9.1. Bridge 29](#_Toc161326139)

[5.9.2. Hub 29](#_Toc161326140)

[5.10. Matter production guide 29](#_Toc161326141)

[5.10.1. CSA Member 29](#_Toc161326142)

[5.10.2. CSA Certification 30](#_Toc161326143)

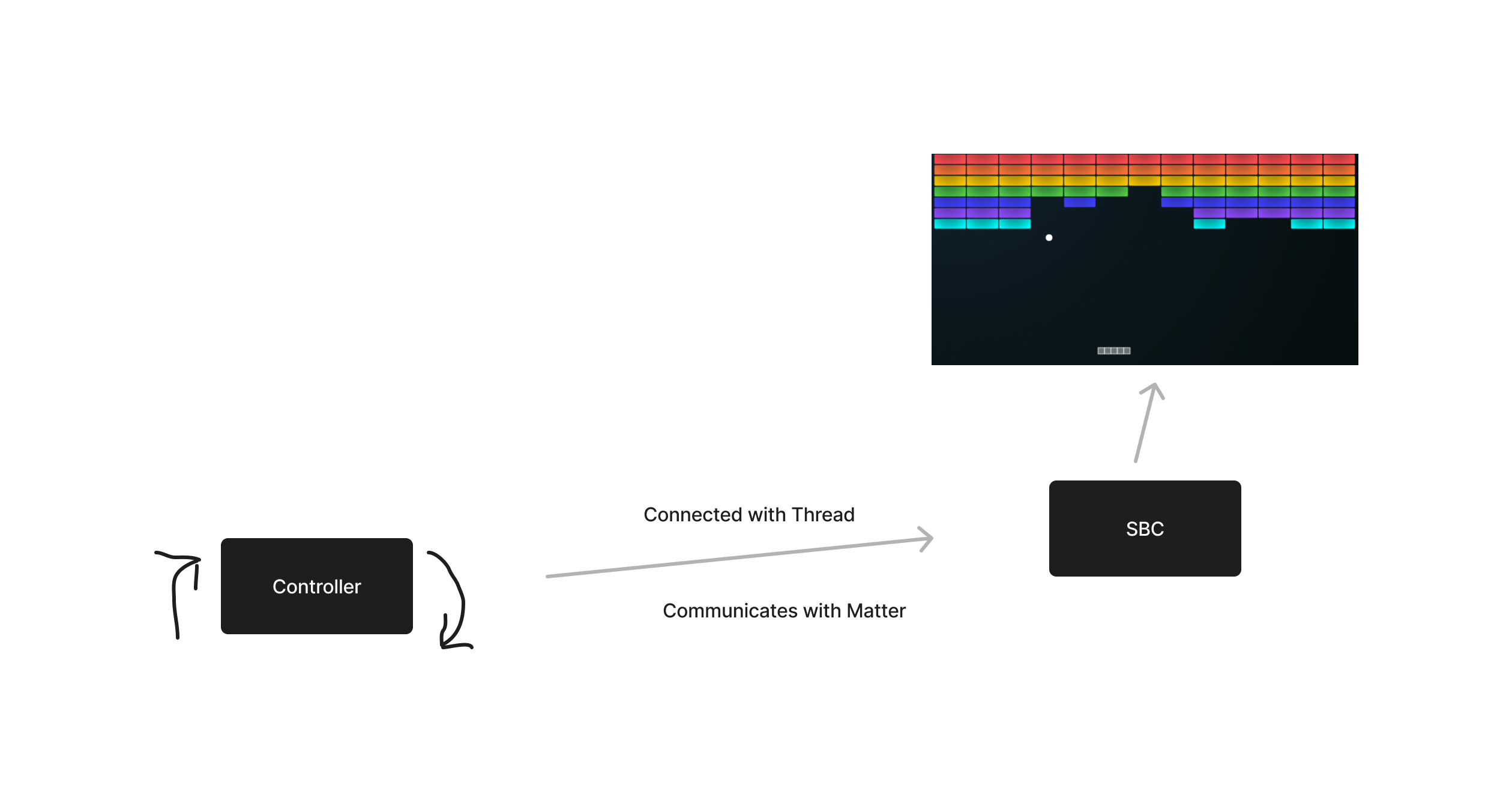
[6. Document Information 31](#_Toc161326144)

[6.1. Version History 31](#_Toc161326145)

[6.1. Related Documents 31](#_Toc161326146)

# Project overview

This project is an interactive breakout game controlled by motion with a wireless controller. The controller is connected to a Thread network and communicates with the Matter protocol. The breakout game is displayed on a tv using a SBC (Single Board Computer) that is also connected to a Thread network.



## Matter protocol

Matter, formerly known as “Project CHIP” (Connected Home over IP) is an open-source, royalty-free connectivity standard designed to make it easier for various smart devices to communicate with each other. It aims to create a unified standard for the Internet of Things devices. A Matter device can connect to a network with Thread, WIFI and Ethernet.

## OpenThread network protocol

OpenThread is an low-power mesh networking protocol that is based on IPv6. OpenThread is a open standard and is built for IoT applications. It uses 6LoWPAN which uses IEEE 802.15.4 (2.4Ghz) wireless protocol with mesh communication.

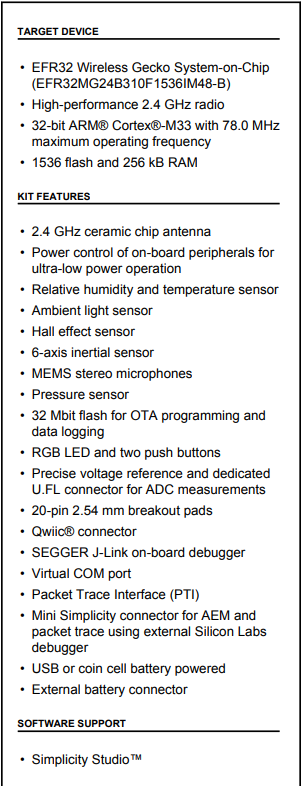
# Game controller

The wireless controller must be equipped with:

* 2.4Ghz wireless controller used for the OpenThread connection
* Accelerometer used to interact with the game (moving the bar)
* Button used for interact with the game (pause, start, etc.)
* A battery to provide power to the controller

## EFR32MG24 - BRD2601B Dev Kit

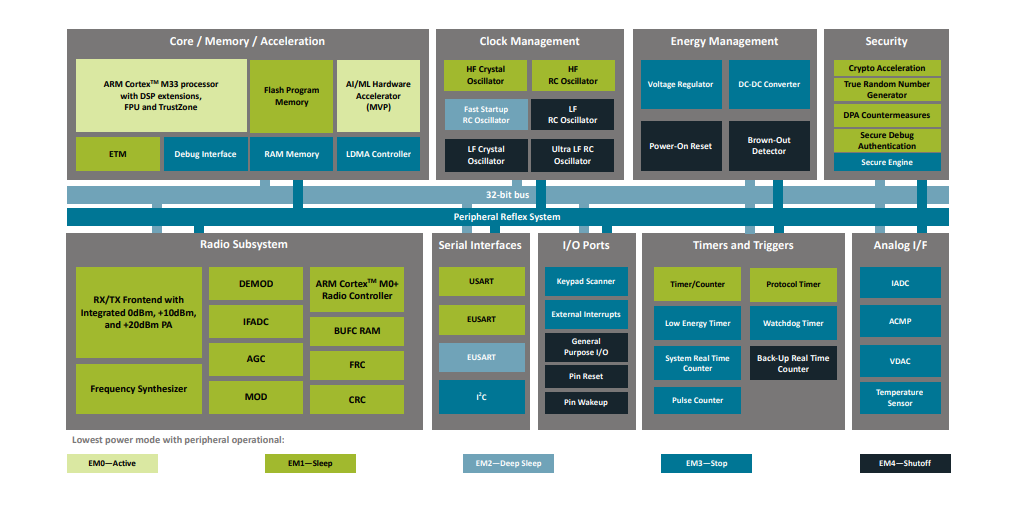
The EFR32MG24 Dev Kit board (BRD2601B) supports al the elements that is necessary for the controller. So there is no need to develop a PCB with al this components.

## EFR32MG24 Wireless SoC

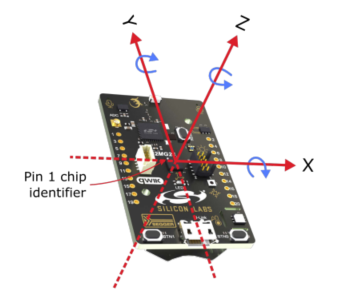
The development board uses the EFR32MG24 wireless SoC (System on a Chip). This SoC is ideal for meshing wireless solutions using Matter and OpenThread. It provides al the built in features that are relevant for this project, like:

* High performance 32-bit 78 MHz Arm Cortex®-M33 processor
* 1536 kB flash and 256 kB of RAM
* High performance 2.4GHz Radio
* OpenThread and Matter support
* Security features for protection against hardware and software attacks
* A wide range of peripherals like I²C, SPI, USART, ADC, Timers, GPIO’s, etc.

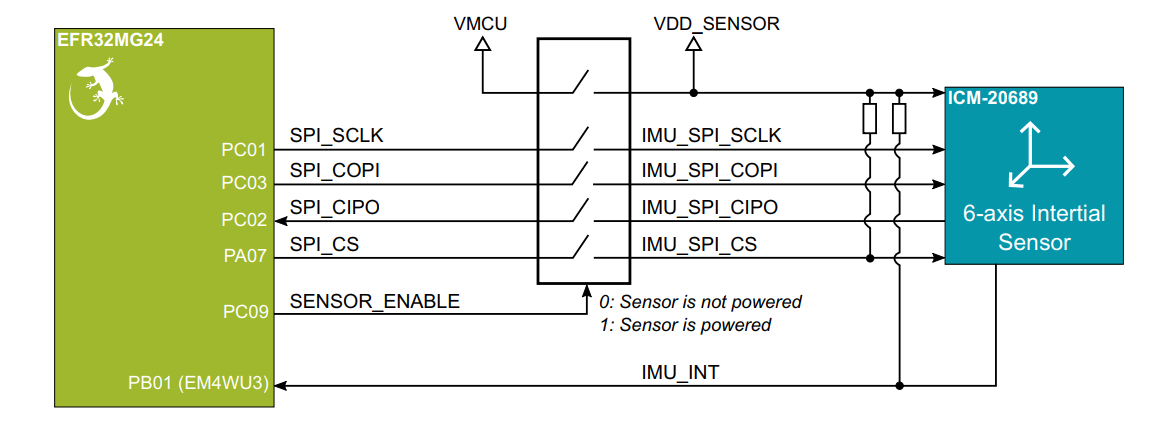


## ICM-20689 6-Axis sensor

The development board contains a 6-axis sensor ICM-20689. This 6-axis sensor combines a 3-axis gyroscope and a 3-axis accelerometer. It detects acceleration and angular rate in the X, Y and Z axes.

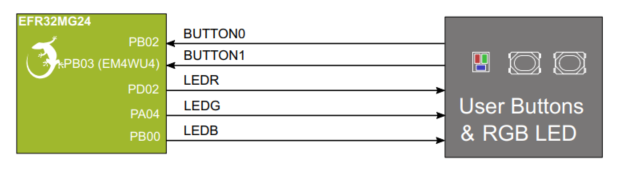


In the development board the sensor is connected an communicates over SPI. The SPI lines are interrupted through a switch to prevent power consumption when not used. Before the sensor can be used in the application it must be enabled by setting PC09 high.

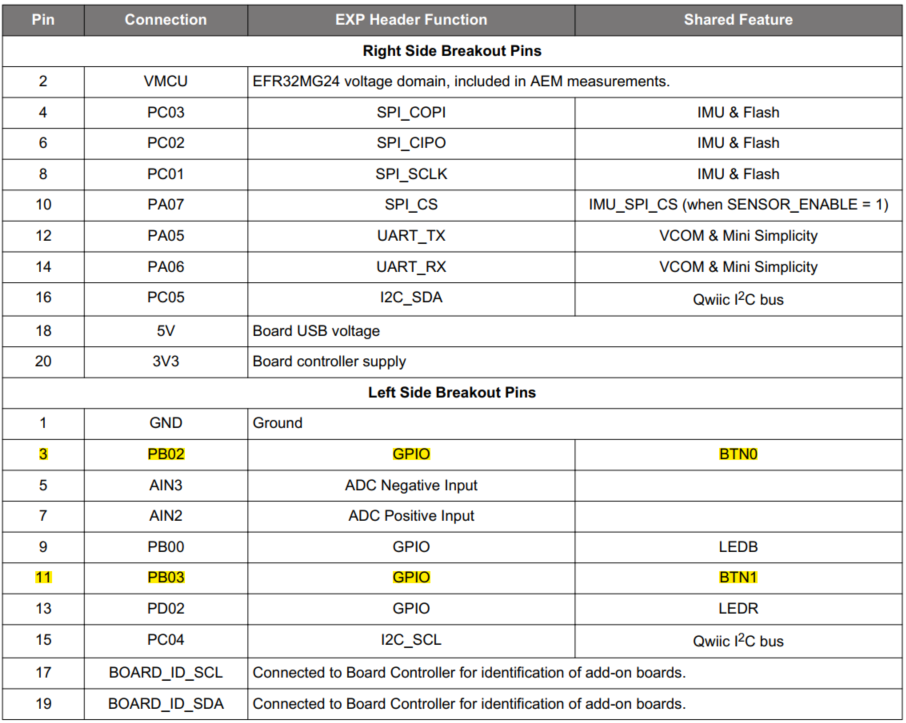


## User Button

The development board has two user buttons, BTN0 and BTN1. They are directly connected to the EFR32MG24 SoC and are debounced by RC filters.



The buttons are connected to the pins PB02 and PB03. The pins are also available on the expansion header. The buttons on the board can be used during the development and later on we can use buttons in a case that are connected to the expansion header.



# (Open) Thread

Thread is a low-power IPv6 mesh networking standard for IoT devices. It is designed for low-power Internet of things devices. The low-power aspect is important for battery-powered smart home devices. However, it’s also low-bandwidth, making it ideal for applications that don’t send a lot of data, like switches or motion sensors.

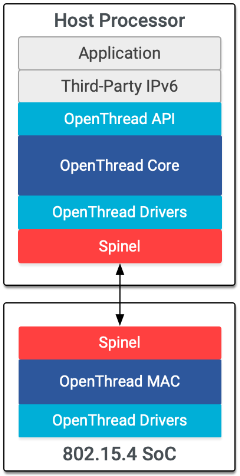
Thread uses the same technology as Zigbee (IEEE 802.15.4) but provides IP connectivity similar to Wi-Fi. Unlike Zigbee, Thread by itself does not allow controlling devices: It is just a communication protocol. To control the Thread devices, a higher-level protocol is required e.g. Matter. Thread devices use the IPv6 standard to communicate both inside and outside the mesh network.

**Thread vs OpenThread**

The Thread protocol specification is available at no cost; however, this requires agreement and continued adherence to an End-User License Agreement (EULA), which states that "Membership in Thread Group is necessary to implement, practice, and ship Thread technology and Thread Group specifications.

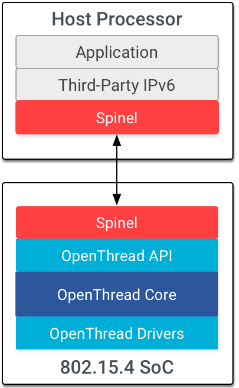
OpenThread released by Google is an open-source implementation of Thread®. Google has released OpenThread to make the networking technology used in Google Nest products more broadly available to developers, in order to accelerate the development of products for the connected home and commercial buildings.

With a narrow platform abstraction layer and a small memory footprint, OpenThread (OT) is highly portable. It supports both System-on-Chip (SoC) and Co-Processor (RCP, NCP) designs.

**RCP**

A OTBR supports a Radio Co-Processor (RCP) design. In this design the core of OpenThread lives on the host processor with only a minimal MAC layer on the processor with the Thread radio. The host processor typically doesn’t sleep in this design, in part to ensure reliability of the Thread network. The advantage of this design is that more resources are available on the more powerful host processor. Communication between the RCP and the host processor is managed by OpenThread Daemon over the Spinel protocol.

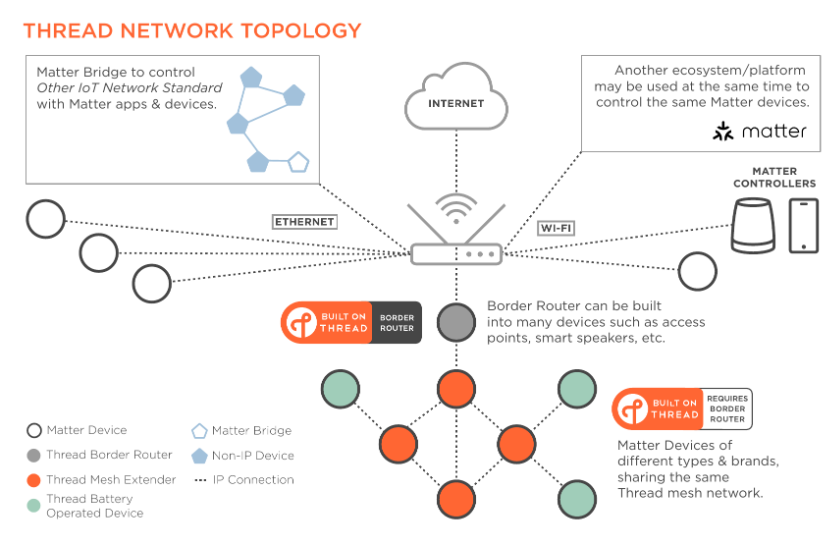
**NCP**



The standard NCP design has Thread features on the SoC and runs the application layer on a host processor, which is typically more capable (but has greater power demands) than the OpenThread device. Communication between the NCP and the host processor is managed by wpantund through a serial interface, typically using SPI or UART, over the Spinel protocol. The benefit of this design is that the higher-power host can sleep while the lower-power OpenThread device remains active to maintain its place in the Thread network. And since the SoC is not tied to the application layer, development and testing of applications is independent of the OpenThread build.

## OpenThread Border Router

A OpenThread Border Router (OTBR) connects a Thread network to other IP-based networks such as Wi-Fi or Ethernet. To communicate outside the mesh network a Border Router is needed. The OTBR routes packets between your local network and the Thread mesh. It does not look at the content of these packets, it just forwards them.



## Rights – OpenThread – Thread

OpenThread released by Google is an open-source implementation of Thread technology. If a company uses OpenThread to build a product, they need to be a member of the Thread Group in order to gain the Intellectual Property (IP) rights to ship Thread products and to complete product certification, which ensures that products using Thread work together effortlessly and securely right out of the box. If developers choose not to join Thread Group and ship products using Thread technology, they are not conferred the IP rights required to practice and ship Thread technology, and may subject themselves to legal action, including but not limited to licensing fees.

# Raspberry Pi 4

In this project, a Raspberry Pi 4 is used as a Single Board Computer. It not only runs the Breakout game but also functions as a Matter hub using the CHIP tool and as a OpenThread Border Router.

The Raspberry Pi 4 is a single-board computer developed by the Raspberry Pi Foundation. It is a popular choice for hobbyists, educators, and professionals due to its affordability, versatility, and ease of use. It is powerful enough for this use case, the most important specs for us is:

* **CPU:** Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
* **RAM:** 4GB LPDDR4-3200 SDRAM
* **Storage:** Micro-SD card slot for loading operating system and data storage
* **Connectivity:**
  + **USB Ports:** It has two USB 3.0 ports and two USB 2.0 ports
  + **Ethernet:** Gigabit Ethernet (RJ45) port for wired networking
  + **Wireless:** Dual-band 802.11ac wireless networking and Bluetooth 5.0
* **Display output:** It has two micro-HDMI ports

# Matter protocol

## Terms

In order to understand this Matter document we need to know the definitions of several terms.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Device | A piece of equipment containing one or more Nodes |
| Node | An addressable entity which supports the Matter protocol stack and has its own Operational Node ID and Node Operational credentials. A Device may host multiple Nodes. |
| Endpoint | A particular component within a Node that is individually addressable. |
| Fabric | A logical collection of communicating Nodes, sharing a common root of trust,and a common distributed configuration state. |
| Cluster |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Root of trust | a concept within Matter that is centered around a certification authority (CA), identified by Root Public Key (Root PK). |
| certification authority (CA) | a device tasked with issuing and assigning Node Operational Certificates (NOCs) or Intermediate Certificate Authority Certificates (ICACs). |
| Node Operational Certificates (NOCs) | are installed during the Matter network commissioning by the commissioner together with Trusted Root CA Certificates. |
|  |  |

## About

Matter is a royalty-free connectivity standard for smart home devices that was announced in December 2019 and officially released in September 2022. It is designed to enable seamless communication and interoperability between smart home devices from different manufacturers, regardless of their underlying technology or platform. Matter was developed by the Connectivity Standards Alliance (CSA), a non-profit organization that promotes interoperability among diverse technology ecosystems. The CSA was formed in 2021 by merging the Zigbee Alliance and the Project Connected Home over IP (CHIP) working group.

## Documents

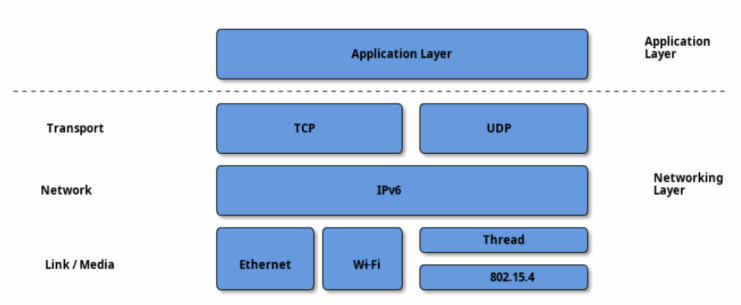
The Matter protocol is specified in Three main documents and can be downloaded from the CSA website. These documents are necessary when developing a Matter product. With every update these documents changes so make sure when developing a product you get the latest documents.

* **Matter-(version)-Core-Specification**In this document you can find all about the features of the protocol, the requirements, technical specifications etc.
* **Matter-(version)-Device-Library-Specification**This document tells which device-types are included in this specific version, it also tells what clusters can and must be used with a specific device-type.
* **Matter-(version)-Application-Cluster-Specification**This document tells all the available clusters in this specific version. It tells what attributes and commands are part of a clusters and which attributes and commands are mandatory or optional for a specific cluster.

## Architecture

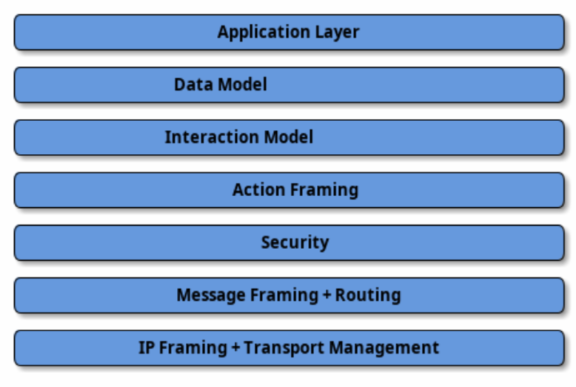
### Network

Matter is a universal IPv6-based communication protocol designed for smart home and the Internet of Things devices. Matter works on the Application Layer of the OSI model.



Matter Protocol Stack

The Matter protocol stack is divided into layers to separate the different responsibilities and introduce a good level of encapsulation amongst the various parts of the protocol stack.



* **Application Layer:** High-order business logic of a device. For example, an application that is focused on lighting might contain logic to handle turning on/off the light as well as its color characteristics.
* **Data Model:** The data layer corresponds to the data and verb elements that help support the functionality of the application. The Application operates on these data structures when there is an intent to interact with the device.
* **Interaction Model:** The Interaction Model layer defines a set of interactions that can be performed between a client and server device. For example, reading or writing attributes on a server device would correspond to application behavior on the device. These interactions operate on the elements defined at the data model layer.
* **Action Framing:** Once an action is constructed using the Interaction Model, it is serialized into a prescribed packed binary format to encode for network transmission.
* **Security:** An encoded action frame is then sent down to the Security Layer to encrypt and sign the payload to ensure that data is secured and authenticated by both sender and receiver of a packet.
* **Message Framing & Routing:** With an interaction encrypted and signed, the Message Layer constructs the payload format with required and optional header fields; which specify the message's properties and some routing information.
* **IP Framing & Transport Management:** After the final payload has been constructed, it is sent to the underlying transport protocol for IP management of the data.

### Network Topology

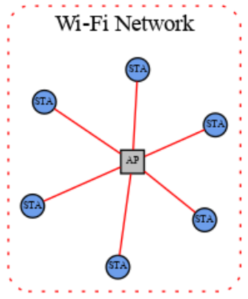
In principle, any IPv6 network is suitable for Matter deployment butt the focus is on three link layer technologies: Ethernet, Wi-Fi and Thread. Matter treats networks as shared resources: it makes no stipulation of exclusive network ownership or access. As a result, it is possible to overlay multiple Matter networks over the same set of constituent IP networks.

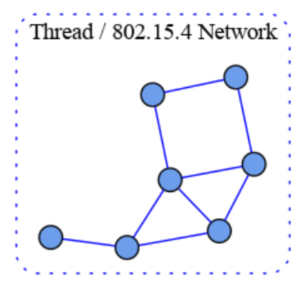
This protocol may operate in the absence of globally routable IPv6 infrastructure. This requirementenables operation in a network disconnected or firewalled from the global Internet. It also enablesdeployment in situations where the Internet Service Provider either does not support IPv6 on consumer premises or where the support proves otherwise limiting.

This protocol supports local communications spanning one or more IPv6 subnets. Canonical networks supporting a fabric may include a Wi-Fi/Ethernet subnet, or one or more low power and lossy network (LLN) subnets. In this version of the specification, Thread is the supported LLN standard.

**Single network**

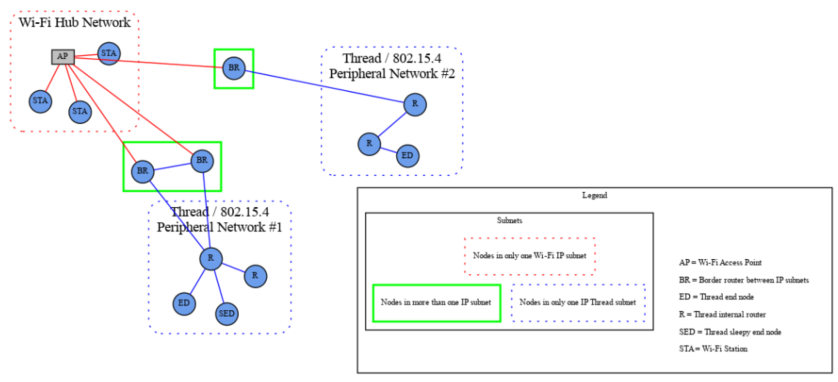
In the single network topology, all Matter devices are connected to a single network. This could by a Thread network, Wi-Fi or Ethernet network. In the case of WiFi/Ethernet, the network could in fact span multiple Wi-Fi and/or Ethernet segments provided that all the segments are bridged at the link layer.





**Star network topology**

The star network topology consists of multiple peripheral networks joined together by a single hub network. The hub network will be the customer’s home network. while the peripheral networks can be of any supported network type. A peripheral network must always be joined directly to the hub network via one or more Border Routers.



In the star network topology any number of peripheral network may be present in a single fabric, including networks of the same type. Nodes may have interfaces onto any network and can communicate directly to other nodes on the same network. If a node needs to cross a network to communicate it must flow through a Border Router.

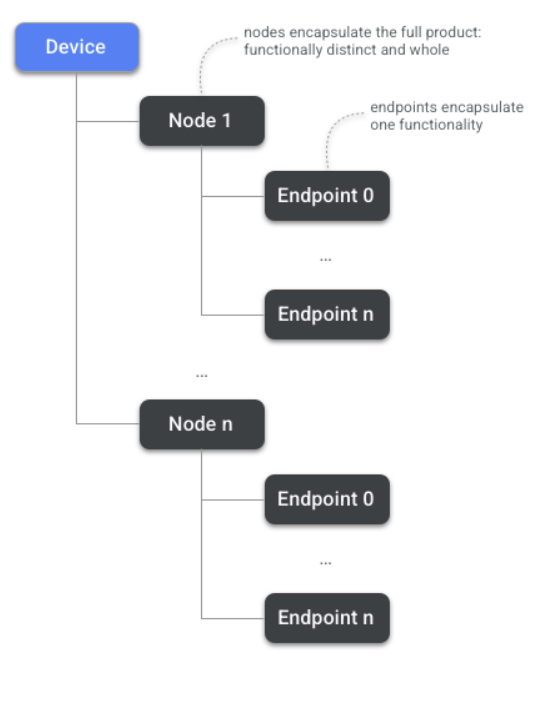
Regardless of the Network topology being used, Matter has a concept of Fabrics. A Matter Fabric is a security domain that contains a collection of nodes. These nodes can be identified and can communicate with each other within the context of that security domain. Each Matter Fabric has a unique Node ID for each node within the fabric and has a unique Fabric ID. Any Matter device can be a part of multiple Matter fabrics, and in turn will have multiple associated Node IDs / Fabric IDs depending on the fabric it is communicating with.

This protocol places a set of requirements on the Border Router. These requirements pertain to

address assignment, route assignment and advertisement, multicast support, and discovery proxying.

## Data Model

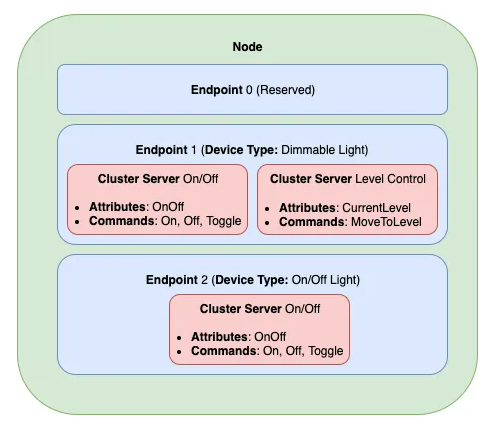
The data model in Matter is a hierarchical modeling of a devices features, including nodes, endpoints, clusters and device types where the node is the highest level data element. A single device can be represented by one or more nodes. An environment where multiple Matter nodes interoperate is referred as a Matter fabric.



### Node

This is a uniquely network addressable entity that exposes some functionality. This is typically a physical device that a user can recognize as a whole device. The role of a node is a set of related behaviors. A node can contain one or more roles including:

* Commissioner: Refers to the process of assigning Fabric credentials to a new device.
* Controller: A node that can control one or more nodes such as a On/Off switch
* Controlee: A node that can be controlled by one or more nodes. Such as an actor. Devices that have the controller role cannot be a controlee.
* OTA Provider: Provides OTA software updates.
* OTA Requestor: Requests OTA software updates.



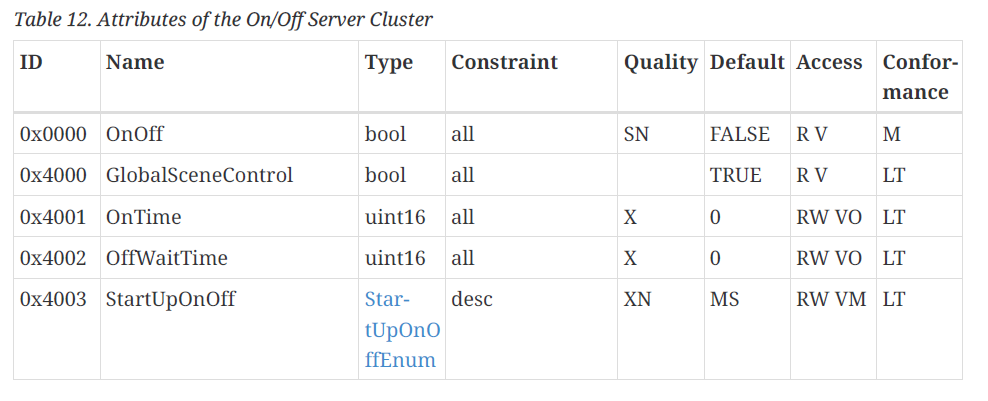
**Endpoint**

Each node has one or more endpoints. A endpoint contain a set functionality’s of a single device. In the example above endpoint 1 is a dimmable light that have the functionality turning on or off AND have a functionality level control, that controls the brightness of the light. Endpoint 2 have only the functionality turning on or off. Note that endpoint 0 is reserved for utility functions.

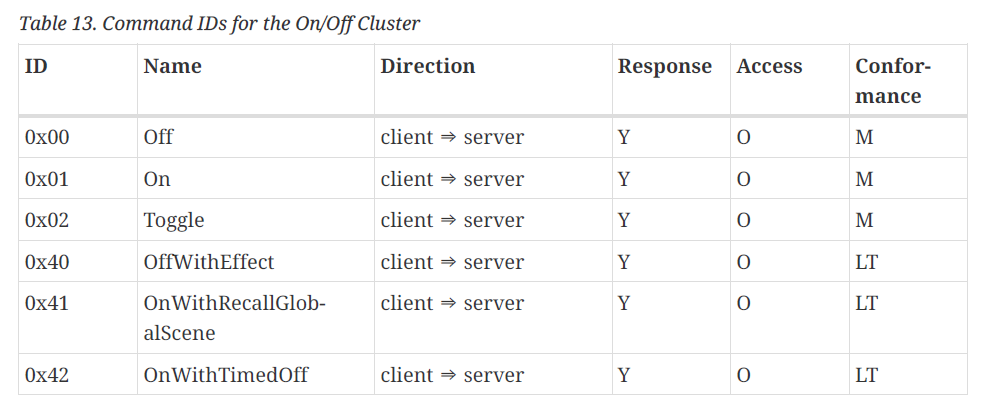
**Clusters**

A cluster groups together commonly used functionality in a reusable building block. Endpoint 1 has two cluster which describes their own functionality. Within the clusters they contain:

* **Attributes**: Attributes are data entities that represent a physical quantity or state. Each attribute is listed in a table with data quality columns: ID, Name, (Data) Type, Constraint, other Quality, Access, Default (value), and Conformance. An attribute also defines its associated semantics and behavior.



* **Commands**: A cluster command provides an ability to invoke a specific behavior on the cluster. A command may have parameters that are associated with it. Each command SHALL be listed in a table with data quality columns: ID, Name, Direction, Response, Access, Conformance. The command table SHALL define the direction of the command as either client to server or server to client. The command table SHALL define the access privileges for each request command or omit the privileges for the default.



* **Events:** Events are a type of attributes that communicate device state changes. They can also be treated as historical data records of something that happened on the device in the past.

The clusters that are supported can be found at the **Matter-(version)-Application-Cluster-Specification** document.

It is possible to make a cluster that is suitable for a specific use case. For example, for a project you need a specific attribute that is not available in the cluster that you going to use. You can extend that cluster so the attribute is available.

**Cluster Client / Server**

A cluster server is stateful and holds attributes, events and commands while a client is stateless and is responsible to initiate interactions with a cluster server.

**Device Type**

A Matter device type is an officially defined collection of requirements for one or more endpoints. Device types are intended to ensure interoperability of different device brands on the market. Each device type definition is composed of the following elements:

* Device type ID
* Device type revision
* One or more mandatory clusters, including each cluster’s minimum revision

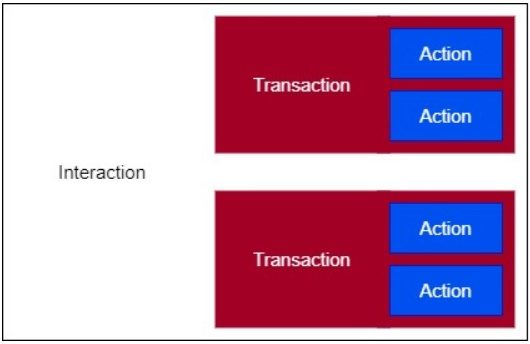
All the device types are defined in the **Matter-(version)-Application-Cluster-Specification.pdf** document.

**Relating Matter to Zigbee**

Ultimately, Matter serves to extend existing protocol stacks to maintain and bolster their architecture for future use. Thus, the Data Model originates from and resembles the Dotdot Architecture Model and Chapter 2 of the Zigbee Cluster Library Specification found here: <https://csa-iot.org/developer-resource/specifications-download-request/>. The Matter Data Model better defines the architecture in the Zigbee Cluster Library while keeping the certifiable cluster specifications.

## Interaction Model

The Matter Interaction Model defines the methods of communication between nodes and serves as the common language for node to node transmission. Node communicate with each other through interactions. Interactions are a sequence of one or more transactions, which in turn are a sequence of actions.



For example, when a client cluster initiate a Read Transaction and requests to read an attribute, the server cluster can respond by giving the attribute. The client request and the server response are separated actions but they are part of the same Read Transaction, these actions and transactions belong to the Read Interaction.

There are four types of interactions:

* **Read Interaction**
* **Write Interaction**
* **Subscribe Interaction**
* **Invoke Interaction**

These types except for Subscribe Interaction consist of one transaction. The Interaction Model supports five types of transactions:

* **Read**: Get attributes and/or events from a server.
* **Write**: Modify attribute values.
* **Invoke**: Invoke cluster commands.
* **Subscribe**: Create subscription for clients to receive periodic updates from servers automatically.
* **Report**: Maintain the subscription for the Subscribe Interaction.

There are several concepts that is important for understanding transactions before explaining the different Interactions.

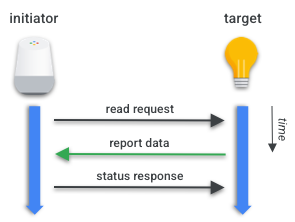
* **Initiators and Targets**: Interactions happen between initiator and target nodes. The initiator starts the transaction and the target responds. Usually the client cluster is the initiator node and de server the target node.
* **Transaction ID**: The transaction ID field must be present in all actions that are part of a transaction to indicate the logical grouping of the actions as part of one transaction. All actions that are part of the same transaction must have the same transaction ID.
* **Groups**: Groups allow an initiator to send an action to multiple targets. This type of communication is known as a groupcast, which leverages IPv6 multicast messages.
* **Paths**: Paths are the location of the attribute, event or command an interaction seeks to access. For example:

****

When groupcasting, a path may include the group or a wildcard operator to address several nodes simultaneously, decreasing the number of actions and thus decreasing the response time of an interaction. Without groupcasting there is a chance of latency between multiple devices reacting to an interaction.

### Read Interaction

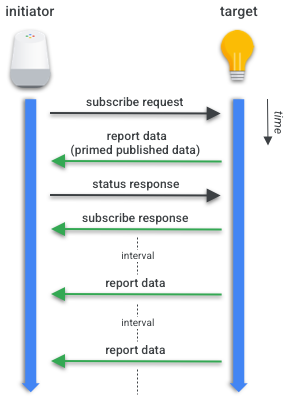
When an initiator wishes to determine the value of one or more attributes or events on a node, an Read Interaction is generated. In this interaction the following actions occur:



* **Read Request:** First action of a Read transaction. The initiator requests a list of the target’s attributes and/or events.
* **Report Data:** Generated in response to the Read Request action. The target sends the requested list of attributes and/or events back along with a suppress response and a subscription ID.
  + Suppress response: a flag that determines whether the status response to this action should be suppressed.
  + Subscription ID: Integer that identifies the subscription transaction, only included if the report is part of a Subscription Transaction.
* **Status Response Action (optional):** Only generated if the suppress response flag is not set. A status Response Action will be generated with a status code:
  + SUCCESS: to continue the interaction
  + INVALID SUBSCRIBTION: if the action is part of a Subscribe interaction and the Subscribtion ID is invalid
  + FAILURE: to terminate the interaction

Read Transactions are restricted to unicast, the Read Request and Report Data actions cannot target groups of nodes because the Status Response Action cannot be generated as a response to a groupcast.

### Subscription Interaction

Subscription is used by a initiator to automatically receive updates of an attribute or event. This creates a relationship between two nodes where the target/publisher periodically generates Report Data Actions to the initiator/subscriber.

A subscribe Request Action contains the following elements:

* **Min Interval Floor:** The minimum interval between reports
* **Max Interval Ceiling:** The maximum interval between reports
* **Attribute Reports:** A list of zero or more of the reported Attributes requested in the Read Action Request.
* **Event Reports:** A list of zero or more reported Events.

After the Subscribe request by the initiator, the target responds with a Report Data Actions containing the first batch of reported data, the pimed published data. Then the initiator acknowledges with a Status Response Action send to the target. Once the Target receives a Status Response Action reporting no errors, it sends a Subscribe Response Action. Then the target will send Report Data Action periodically at the negotiated interval until the subscription is lost or cancelled.

Subscription interactions have some restrictions:

* The subscribe Request and Response Action are Unicast-only
* Report Data Actions in the same Subscription Interaction must have the same Subscription ID.
* If the subscriber does not receiver a Report Data Action within the maximum interval between Actions, the subscription will be terminated.

### Write Interaction

The Write Interaction is generated when a initiator wants to modify values of one or more attributes located on one or more nodes. The initiator has the option to use a Timed or a Untimed Write Transaction.

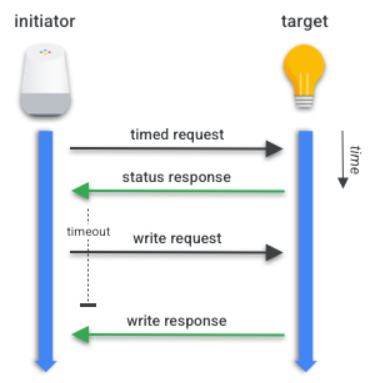
**Untimed Write Transaction**

In the Untimed Write Transaction there are two actions:

* **Write Request Action:**A Write Transaction works similar to the Read Request Action. The initiator provides the target with:
  + **Write Request:** A list of one or more tuples containing Path and data.
  + **Timed Request:** A flag that indicates whether this action is part of a Timed Write Transaction.
  + **Suppress Response:** A flag that indicates whether the Response Status Action should be suppressed
* **Write Response Action:**When the target receives the Write Request Action it will respond with the Write Response Action containing:
  + **Write Response:** A list of paths and error codes every Write Request send on the Write Request Action.

The Write Request Action may be a groupcast, but in this case the Suppress Response flag must be set. The rationale is that otherwise the network might be flooded by simultaneous responses from every member of a group. To enable this behavior, the Path used in the Write Requests list may contain Groups and alternatively they may contain wildcards, but only on the Endpoint field.

**Timed Write Transaction**

In this transaction there where added a few steps to the untimed write transaction.

* **Timed Requests Action:**When the initiator starting a transaction, the action contains:
  + **Timeout:** how many milliseconds this transaction may remain open. During this period the next action sent by the Initiator will be considered valid.

When the target receives the Timed Request Action, he must acknowledge with a Status Response Action. When the initiator receives a response with no errors, it will send a Write Request Action.

The rest of the actions are the same as the Untimed Write Transaction.

## Security

To protect the Matter fabric, there are several security features implemented to make sure that only trustworthy device can join the network and protecting the messages that are exchanged between the fabric nodes.

### Session establishment

This is used the exchange encryption keys that are required for a secure communication between nodes. It also involves mutual node authentication, which assures both nodes that they initiate communication with a trusted peer.

There are two session establishment methods available:

* **PASE**: Passcode-Authenticated Session Establishment  
  When using PASE, both nodes share the same secret in the form of 8-digit passcode. The shared secret is used by the SPAKE2+ algorithm to ensure a safe exchange of keys over non-secure channel. This process takes place when commissioning the device.
* **CASE**: Certificate-Authenticated Session Establishment  
  When using CASE, both nodes own Node Operational Certificates that chain back to the same root of trust. The NOCs are used by the SIGMA algorithm to ensure a mutual node authentication and a safe exchange of keys over non-secure channel. This process takes place while establishing the secured communication between nodes that are already commissioned.

### Message confidentiality and integrity

After exchanging the keys and establishing secure channel, the 128-bit AES-CCM algorithm is used to provide both confidentiality and integrity of exchanged messages.

A Matter message consists of the following elements:

* **Message Header:** Carries session and transport-related information.
* **Protocol Header:** Describes semantics of the message.
* **Payload:** Actual protocol-specific content of the message.

While the AES-CCM algorithm ensures the integrity of all three elements, only Protocol Header and Payload get encrypted. This is because Message Header contains fields, such as Security Flags and Message Counter, which are used to calculate the AES-CCM Nonce that is necessary to decrypt the remaining part of a message.

## Commissioning

When we want to add a device to a Matter network, we first need to commission it. You can think of it as the initial pairing. Although Matter communication in general uses Wi-Fi, Ethernet and Thread, commissioning can be performed over Bluetooth Low Energy (BLE).

The commissioning process takes place between a commissioner and a commissionee. Where the commissioner is the controller that is carrying out of the commissioning and the commissionee is the device that wants to join the fabric.

### Prerequisites

The controller must get onboarding payload from the commissionee to start the procedure. The data payload includes the following information, among other things:

* 16-bit Vendor ID and 16-bit Product ID
* 12-bit device discriminator
* 27-bit setup passcode
* 8-bit Discovery Capabilities Bitmask

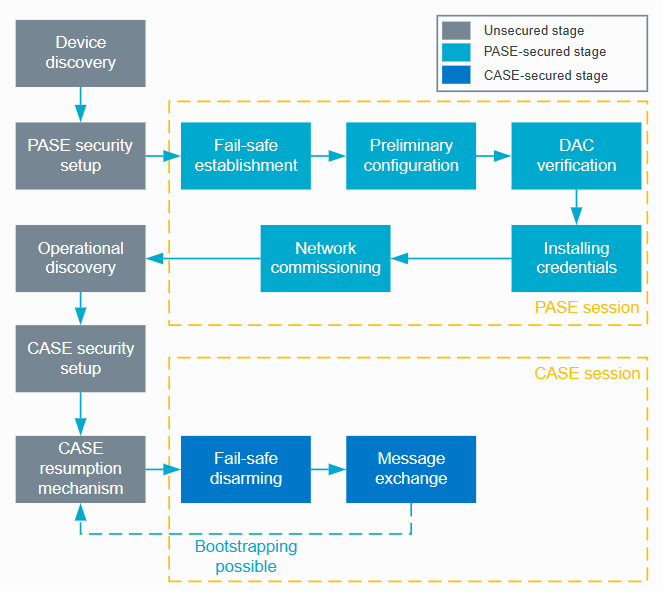
### Onboarding payload

The Onboarding Payload is the information used by the Commissioner to ensure interoperability between commissioners and devices. The commissioner can collect this information in different ways:

* **QR Code:** which you can scan using a mobile device with the ecosystem of your choice.
* **QR Code Payload:** which is an alphanumeric code that you can use in command-line tools. For testing purposes, it can be printed to the UART console or be shared using an NFC tag. This code is represented visually by the QR Code.
* **Manual Pairing Code:** provides the onboarding information as a sequence of digits that can be used with most Matter commissioners.

### Commissioning steps

The commissioning process consists of steps shown in the figure below.



**Device discovery**

The commissioner discovers devices that can be commissioned onto the network. The commissionee needs to advertise their presence so the commissioner knows about their existence. This can happen using the following methods:

* **Bluetooth Low Energy:** This method is used especially if the node is being added to its first Matter fabric.
* **DNS-SD:** This method is commonly used if the node is connected to Ethernet or is already a member of a Wi-Fi or Thread network.

**PASE security setup**

The commissioner runs the Passcode-Authenticated Session Establishment (PASE) protocol, which is exclusive to the commissioning process. This protocol is used to establish the first session between devices that take part in commissioning. The session is established with a passcode provided out-of-band and that is used to derive encryption keys. This passcode is known only to the commissioner and the commissionee.

There can be only one ongoing PASE sessions at a time.

**Fail-safe establishment**

The commissioner requests the commissionee to back up its original configuration. The fail-safe acts as a back-up, but it also starts a timer that sets a limit for the whole commissioning process. The timer is disabled with the disarming of the fail-safe at the end of commissioning.

**Preliminary configuration**

The commissioner reads the Basic Information Cluster attributes of the commissionee and its device type. It then configures the commissionee with regulatory information, such as location and country, and the current UTC time.

**Device Attestation Certificate verification**

The commissioner checks whether the commissionee is a certified Matter device. As part of this verification, the commissioner generates a random 32-bit attestation nonce and sends it to the commissionee, who should return the signed attestation information that includes the nonce. Usage of a nonce prevents replay attacks against commissioners. The commissioner then validates the attestation information.

The verification succeeds if the device is able to prove the validity and ownership of the mandatory Matter Device Attestation elements. If the validity and ownership cannot be proven, the verification fails. The commissioner can then either terminate or continue the commissioning procedure.

**Installing credentials**

The commissioner installs Node Operational Certificate (NOC) and Operational ID on the commissionee. The commissionee becomes the new node of the Matter fabric. The node is identified by a tuple consisting of the Root PK, Fabric ID, and Node ID. (While the fabric is identified by a tuple consisting of the Root PK and the Fabric ID.)

**Network commissioning**

The commissioner provisions the commissionee node with the operational network credentials, either Wi-Fi or Thread, and requests the commissionee to connect to the network.

**Operational discovery**

The commissioner discovers the commissionee node on the operational network using DNS-SD. This way, the commissioner learns the IP address of the node.

**CASE security setup**

The commissioner and the node use the Certificate-Authenticated Session Establishment (CASE) protocol to establish secure communication. The CASE protocol is in charge of exchanging NOCs to set up a session secured with a new pair of keys. The CASE connection is reset each time a device breaks the connection.

**Fail-safe disarming**

The commissioner requests the commissionee node to remove the stored configuration backup. This also stops the fail-safe timer.

**Message exchange**

The commissioner and the commissionee start exchanging AES-encrypted messages on the operational network.

## Matter Hub & Bridge

### Bridge

A Matter bridge act as a translator, it allows non Matter devices to connect and work with Matter devices in a fabric. With a Matter bridge Zigbee and/or Z-Wave devices can work with Matter devices, this allows more flexibility in an existing smart-home.

### Hub

A Matter hub essentially serves as the controller of the network, necessary for managing your Matter devices. It typically isn't tied to a specific ecosystem, allowing users the freedom to choose their preferred ecosystem. Additionally, the Matter Hub often functions as a Thread Border Router but not always.

## Matter production guide

To develop a Matter end product, this topic lists the prerequisites and next steps to facilitate your production journey through Matter.

### CSA Member

Associated membership is necessary to get a product certified by a CSA-approved testing facility. As a member you will receive membership perks like:

* Official resources to assist you in developing Matter products.
* Authorization to contribute to the [Matter Github repository](https://github.com/project-chip/connectedhomeip).
* Once approved, CSA will reserve a unique Vendor ID (VID) chosen by your organization. This VID will be needed to provision your device. Your unique VID will be added to the [CSA Distributed Compliance Ledger](https://webui.dcl.csa-iot.org/) (DCL).
* Matter [Certification tool access](https://csa-iot.org/certification/tools/certification-tool/). This allows you to evaluate your product for certification before the official certification process

Become a member at [CSA Membership](https://csa-iot.org/become-member/). You can see a list of the different memberships offered at CSA.

### CSA Certification

Once a product is ready for production, the product must be certified by a CSA-approved testing facility to certify compliance with the Matter Standard. Check the [CSA Certification process](https://csa-iot.org/certification/why-certify/) for more information. For additional resources on the certification process, refer to the [Certifying your Matter Product](https://community.silabs.com/s/article/Certifying-your-Matter-Product-an-Overview?language=en_US) article by Silicon Labs.

# Document Information

## Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Description** |
| 22/09/2023 | BGY | 0.1 | Project overview and wireless controller description |
| 14/04/2024 | BGY | 0.2 | Added Thread and Matter fundamentals |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Related Documents

|  |  |
| --- | --- |
| **Ref** | **Description** |
| Matter-1.2-Core-Specification.pdf | Full specification document of the Matter protocol. |
|  |  |
| Matter-1.2-Application-Cluster-Specification.pdf | Supported application cluster specification. |
|  |  |
| Matter-1.2-Device-Library-Specification.pdf | All the device types are defined in this document. |
|  |  |