ZIPATO MQTT DOCUMENTATION

1.0.0	Initial version
1.1.0	Added more details to generic network actions. Verbose description of Z-Wave inclusion and exclusion procedures. Added networks/status topic description
1.2.0	Added more description to all Z-Wave specific network actions Changed names of all Z-Wave specific network actions to make them more consistent with the rest of the api Documented actions for setting and getting Z-Wave specific device configuration Fixed wrong response topic for attributes
1.2.1	Documented name changes for certain Z-Wave specific network actions Cleaned up JSON examples
1.2.2	Add missing states to GET_CONF
1.3.0	Added device state transition description and state transition diagram
1.4.0	Added binary sensor endpoints `epId` table
1.5.0	Added Z-Wave manufacturer specific device identification tables
1.6.0	Added invoke interface
1.7.0	Added Z-Wave protocol data backup/restore action descriptions Added general backup procedure description\

1.8.0	Added set/get wakeup interval actions. Fixed wrong parameter name for set/get configuration actions. Added table of contents. Fixed manufacturer specific id's for some devices.
1.9.0	Documented possible failures when calling CHECK_FAILED_NODES
2.0	Added new invoke actions for Z-Wave network Added `getValue` action to attribute topic
2.1	Clarified FAILED status for Z-Wave specific SET/GET configuration commands
2.2	Added `secureInclusionDisabled` parameter
2.3	Added description for replace failed node and learn mode Z-Wave network actions
2.4	Clarifications to invoke interface
2.5	Fixed wrong status messages
2.6	Add timeout parameter for exclusion
2.7	Add timeout parameter for inclusion
2.8	Add device model introduction
2.9	Add Z-Wave OTA firmware update action
3.0	Add MQTT KeepAlive description
3.1	Updated device topic description

Table of contents

Zipato Device Model introduction

Zipato device model is designed to be a generic abstraction layer that can be used to model various communication protocols and devices.
The device model is a hierarchy of entities that represent various aspects of any given protocol or device.
Zipato Device Model in general is comprised of several types of entities.
Each entity in the device model is uniquely identifiable with an UUID.
Here is a simple overview of the device model hierarchy:

- Network(s)
    - Device(s)
        - Endpoint(s)
            - Cluster(s)
                - Attribute(s)

Network
Networks usually represent a particular protocol, like Z-Wave or Zigbee for example. Networks can have one or more devices. Almost all networks have a few common actions, like adding a device, removing a device or resetting a device. A network can also have network specific actions.

Device
A device represents an actual physical device, a wall plug or a door sensor for example. A device typically has one or more functions. Networks can have one or more devices.

Endpoint
Endpoints are used for “partitioning” various functions and features of a device according to its features and limitations of the device model itself. There are no general rules on how endpoints should be used or what should they represent. However, each device usually has at least one endpoint, or several.

Cluster
A cluster represents a particular device function, like a reed sensor of a door sensor, or a relay of a wall switch for example. An endpoint can contain one or more clusters that represent functions of a particular device.
One important constraint of the device model is that an endpoint cannot contain more than one cluster of a particular type.
For example, if a device has two relays, clusters representing these relays will of course be of the same type. But they cannot be under the same endpoint, each has to be under its own endpoint.

Attribute
Attributes represent one or several states or values of a particular device function.
For example, reed sensor state of a door sensor or relay state of a wall plug can be modelled with a simple boolean value; Both can be either on or off.
That means that clusters that model these functions(simple binary sensor and simple binary actuator in this example) will have a single attribute that holds the current state of these functions.

Attributes can be readable, writable or both. For example an attribute of a cluster that models a simple binary sensor will be readable, but not writable. It makes no sense to “write” the state of a sensor device.
An attribute of a cluster that models simple binary actuators, like a relay of a wall plug, will be both readable and writable; You can check its state and you can change the state.

Attribute definitions
Attributes model, or “hold” so to speak, the current state or value of a particular function of a device. Certain device functions have additional associated data.
For example, a wall plug might have a meter function, one that can measure current power consumption and voltage.
Both of these functions have units that describe their values. An attribute definition is used to hold additional data about a certain device function. Each attribute has an associated attribute definition.

Box status topics

Request Topic: N/A,
ResponseTopic: /local/ha/bridges/zipato/conn_status
Payload type: String, “online” or “offline”
Description: Last will configuration of Mqtt connection. sent automatically when client establishes connection and when connection is lost

---

Request Topic: /local/ha/bridges/zipato/box/info
Response Topic: request/box/info
Request Topic Payload: N/A
Response Type: JSON
Response example:

```
{
	"mqttApiVersion": "1.0.0",
	"boxSerial": "boxSerial",
	"boxFw": "boxFw"
}
```

Description: Sent automatically when client establishes connection or on request via request topic. Contains the box serial number, firmware version and current mqtt api version.

Request Topic: N/A
Response Topic: /local/ha/bridges/zipato/box/messages
Request Topic Payload: N/A
Response type: JSON
Response Example:

```
{
	{"SYSTEM":"STARTED"}
}
```

Description: Various unsolicited system messages.

Request Topic: N/A
Response Topic: /local/ha/bridges/zipato/box/KeepAliveEvents
Request Topic Payload: N/A
Response Type: JSON
Response example:
{"className":"com.zipato.event.MQTTKeepAliveEvent"}
Description: Keep alive ticker. By default it emits a tick every 10 seconds.

Box actions

Request box synchronisation
Request Topic: request/box/sync
Request Topic Payload: N/A
Response/Result topic: /local/ha/bridges/zipato/box/messages
Response Type: JSON
Response Example:
{"progress":"complete","command":"LOAD_ALL"}
Description: Start cloud synchronisation. At the moment this is still a necessary step after successful device discovery.

Network topics

Request Topic: request/networks/list
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/networks/list
Response Type: JSON
Response Example:


```
[{
        "link": "/networks/df214ae6-653e-4d3d-a195-87c241799e82",
        "name": "IP",
        "uuid": "df214ae6-653e-4d3d-a195-87c241799e82"
}, {
        "link": "/networks/de6a9aa9-5555-4a99-8d46-54be03573569",
        "name": "IP backup",
        "uuid": "de6a9aa9-5555-4a99-8d46-54be03573569"
}, {
        "link": "/networks/a7c13a52-9e50-434c-b0e9-fa3b009163d5",
        "name": "Mqtt",
        "uuid": "a7c13a52-9e50-434c-b0e9-fa3b009163d5"
}, {
        "link": "/networks/f5f264a0-0473-4c10-92c7-0fcfd1cebd5c",
        "name": "Zwave",
        "uuid": "f5f264a0-0473-4c10-92c7-0fcfd1cebd5c"
}]
```


Description: A list of available networks on the current box.

Request Topic: request/networks/$UUID/info
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/networks/$UUID/info
Response Type: JSON
Description: Various network specific data and current status. Also contains a list of devices in that network.
Response Example:


```
{
        "link": "networks/a7c13a52-9e50-434c-b0e9-fa3b009163d5",
        "config": {
                "className": "com.zipato.network.mqtt.MqttNetwork",
                "uuid": "a7c13a52-9e50-434c-b0e9-fa3b009163d5",
                "name": "Mqtt",
                "cv": 0,
                "sv": 0,
                "deleted": false,
                "nd": true,
                "tags": null,
                "order": null,
```

```
                "param": null
        },
        "devices": [],
        "state": {
                "network": "a7c13a52-9e50-434c-b0e9-fa3b009163d5",
                "discovery": false,
                "online": false,
                "trouble": false
        },
        "stateTimestamp": "2016-04-04T19:44:59Z",
        "templateId": null,
        "uuid": "a7c13a52-9e50-434c-b0e9-fa3b009163d5"
}
```

Request Topic: N/A
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/networks/$UUID/messages
Response Type: JSON
Description: Unsolicited network messages that occur as a result of some network actions like network initialization, device discovery, device configuration or device deletion for example.
Response Example:


```
{
{"secondary":false,"sisNodeId":1,"inclusionCtrl":false,"final":false,"primarySis":true,"zNodeId":1,"type":"INIT","homeId":3782768885,"
network":"ZWAVE","status":"DONE"}
}
```

Request Topic: N/A
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/networks/$UUID/status
Response Type: JSON
Description: Unsolicited network messages that are usually sent when a network goes online or offline. A network usually comes online after it is initialized. A network usually goes offline if its initialization fails.

Generic network actions

All network action response payloads are of course in json format. **Every** network action response json object has a a couple of key properties:

- `type`: this property contains the name of the current action.
- `network`: this property contains the name of the network on which this particular action is executing

- `status`: Status of the current action
- `final`: This boolean property is used to denote the final response of the current action. A response that has the final field set to `true` marks the end of that action

Certain network actions will have more properties in their response objects.
A general limitation on almost all networks is that they can execute **one action at a time**.

Start device discovery
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:

```
{
        "className": "com.zipato.event.CommandEvent",
        "command": "DISCOVERY_ON"
}
```

Optional properties:

- `secureInclusionDisabled` - setting this property to `true` will disable Z-Wave secure inclusion

Description: Generic action to start discovery on any network that supports device discovery.

Stop device discovery
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:

```
{
        "className": "com.zipato.event.CommandEvent",
        "command": "DISCOVERY_OFF"
}
```

Description: Generic action to stop discovery on any network that supports device discovery. On some networks, like Z-Wave for example, this command is also used to stop any current ongoing network action.

Z-Wave specific network actions

Certain Z-Wave specific network actions names have been changed in firmware version 1.0.9.1:

- checkFailedNodes -> CHECK_FAILED_NODES
- getProtocolStatus -> GET_PROTOCOL_STATUS
- enableRadio -> ENABLE_RADIO
- disableRadio -> DISABLE_RADIO

Start device exclusion
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:

```
{
        "className": "com.zipato.event.CommandEvent",
        "command": "DELETE_DEVICE_ZW
}
```

Description: Starts the exclusion procedure.
Optional request payload parameters:

- `timeout`: exclusion timeout, in seconds. Exclusion will be stopped if a device is not found in this time interval

Response object properties:

- `type`: “REMOVE”
- `Network`: “ZWAVE”
- `nodeId`: the node id of the removed node. This property will be set to 0 if a node from a different network was excluded
- `device`: UUID of the removed device. This property is set if a known device was excluded
- `status`:
    - STARTED: Z-Wave module now operating in exclusion mode
    - NODE_FOUND: Z-Wave module has detected a device running in exclusion mode
    - SUCCESSFUL: Exclusion complete.
    - TIMEOUT: No device found for exclusion in 45 seconds, exclusion mode turned off
    - UNSUCCESSFUL: Exclusion failed, cause of failure set in `message` property
- `Message`: cause of the exclusion failure

Start device inclusion
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:

```
{
        "className": "com.zipato.event.CommandEvent",
        "command": "DISCOVERY_ON"
}
```

Description: Starts the inclusion procedure.

Optional request payload parameters:

- `timeout`: inclusion timeout, in seconds. Inclusion will be stopped if a device is not found in this time interval

Response object properties:

- `type`: “INCLUSION”
- `Network`: “ZWAVE”
- `status`:
    - STARTED: Z-Wave module now operating in inclusion mode
    - DEVICE_FOUND: Z-Wave module has detected a device running in inclusion mode
    - SUCCESSFUL: inclusion complete.
    - NODE_ALREADY_INCLUDED: Self explanatory
    - TIMEOUT: No device found for inclusion in 60 seconds, inclusion mode turned off
    - FAILED: Inclusion failed, cause of failure set in `message` property
- `Message`: cause of the exclusion failure

A successful inclusion is always followed by a secure inclusion procedure if the device supports the SECURITY command class:

Secure inclusion procedure response object properties:

- `type`: “SECURE_INCLUSION”
- `Network`: “ZWAVE”
- `Status`:
    - DONE: Secure inclusion successful
    - FAILED: Secure inclusion failed, cause of failure set in `message` property
    - TIMEOUT: Secure inclusion timed out cause of timeout set in `reason` property
- `reason/message:` cause of timeout or failure

Secure inclusion may sometimes fail. The state of the device after that is mostly unknown. The device might be fully operational or certain features might not work. It is advised to exclude the device and include it again if secure inclusion fails.

A successful inclusion is also always followed by a device configuration procedure that interviews/configures the newly joined device:

Device configuration response object properties:

- `type`: “REDISCOVERY”
- `Network`: “ZWAVE”
- `Status`:
    - SUCCESSFUL: Device configuration successful

    - UNSUCCESSFUL: Device configuration failed, cause of failure set in `reason` property
    - INVALID_RESPONSE: Usually indicates a fatal exception during device configuration
    - TIMEOUT: Device configuration timed out, cause of timeout set in `reason` property
- `reason/message:` cause of timeout or failure

Start failed node check procedure.
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"NETWORK_ACTION", "action": "CHECK_FAILED_NODES"}
Description: Starts the failed node check procedure.
Device configuration response object properties:

- `action`: “CHECK_FAILED_NODES”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - NO_FAILED_NODES
    - FAILED_NODE_FOUND: at least one failed node was found
    - FAILED: reason of failure set in `reason` property
- `Reason`: cause of failure. A failure is always a missing response from the Z-Wave module. The field will indicate which response was not sent:
    - Missing list of failed nodes
    - Missing response to a failed node check

Disable Z-Wave radio
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"NETWORK_ACTION", "action": "DISABLE_RADIO"}
Description: Disables Z-Wave RF transmitter and blocks all RF communication.
Device configuration response object properties:

- `action`: “DISABLE_RADIO”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED

    - DONE

Enable Z-Wave radio
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"NETWORK_ACTION", "action": "ENABLE_RADIO"}
Description: Enables Z-Wave radio and unblocks RF communication.
Device configuration response object properties:

- `action`: “ENABLE_RADIO”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - DONE

Get protocol status
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"NETWORK_ACTION", "action": "GET_PROTOCOL_STATUS"}
Description: Retrieves current protocol status from the Z-Wave module.
Device configuration response object properties:

- `action`: “GET_PROTOCOL_STATUS”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - ZW_PROTOCOL_IS_IDLE
    - ZW_PROTOCOL_STATUS_ROUTING
    - ZW_PROTOCOL_STATUS_SUC
    - UNKNOWN_RETURN_VALUE
    - FAILED: reason of failure set in `reason` property
- `Reason`: cause of failure

Get RF power level
Request Topic: request/networks/UUID/info
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload: N/A
Description: Current RF power level is contained within network specific information retrieved via this request topic

---

Hard reset Z-Wave module
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", "zwCmd": "zwHardReset"}
Description: Hard reset Z-Wave module by calling ZW_SetDefault(). Deletes all node and routing data from the module and resets the controller node id to 1 and home id to a random value
Device configuration response object properties:

- `type`: “HARD_RESET”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - DONE
    - FAILED: reason of failure set in `reason` property
- `Reason`: cause of failure

Set device specific configuration
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/devices/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON",
"paramId":"3","value":1,"zwDevUUID":"e0a52321-88d1-4f6a-9b83-9aba6afdcf27","zwCmd":"ZwConfHandle
r_SET_CONF"}
Description: Set device specific configuration parameters via configuration command class commands.
Response object properties:

- `type`: “SET_CONF”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - WAITING_WAKEUP_NOTIFICATION: Waiting for the device to wake up
    - TIMEOUT_WAKEUP_NOTIFICATION: Timed out while waiting for the device to wake up (30 seconds)
    - DONE
    - FAILED: Indicates failed transmission for SET command and missing report for GET command
- `paramId`: current configuration parameter
- `value`: value of the current configuration parameter

Get device specific configuration
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/devices/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", "paramId":"3",
"zwDevUUID":"e0a52321-88d1-4f6a-9b83-9aba6afdcf27","zwCmd":"ZwConfHandler_GET_CONF"}
Description: Get device specific configuration parameters via configuration command class commands.
Response object properties:

- `type`: “GET_CONF”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - WAITING_WAKEUP_NOTIFICATION: Waiting for the device to wake up
    - TIMEOUT_WAKEUP_NOTIFICATION: Timed out while waiting for the device to
    - DONE
    - FAILED: reason of failure set in `reason` property
- `paramid`: current configuration parameter
- `value`: value of the current configuration parameter
- `Reason`: cause of failure

Backup Z-Wave module protocol data
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/devices/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", "zwCmd": "zwBackup"}
Description: Backup Z-Wave module protocol data to a file. Data will be written to `$ZIPATO_HOME/storage/zwaveProtocolDataBackup.bin` file.
Response object properties:

- `type`: “BACKUP”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - DONE
    - FAILED: reason of failure set in `reason` property
- `Reason`: cause of failure

Restore Z-Wave module protocol data
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/devices/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", "zwCmd": "zwRestore"}
Description: Restore Z-Wave module protocol data from a file. Data will be read from `$ZIPATO_HOME/storage/zwaveProtocolDataBackup.bin` file.
Response object properties:

- `type`: “RESTORE”
- `Network`: “ZWAVE”
- `Status`:
    - STARTED
    - DONE
    - FAILED: reason of failure set in `reason` property
- `Reason`: cause of failure

Remove failed node
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", “zwDevUUID”: “UUID”, "zwCmd": "zwRemoveFailedNode"}
Description: Starts the exclusion procedure.
Response object properties:

- `type`: “REMOVE_FAILED_NODE”
- `Network`: “ZWAVE”
- `status`:
    - STARTED: Remove failed node procedure started
    - CANNOT_REMOVE: Node is reachable, it cannot be removed
    - DONE: Failed node removed
    - FAILED: Failed to remove node, cause of failure set in `reason` property
- `reason`: cause of the failed node removal failure

Replace failed node
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", “zwDevUUID”: “UUID”, "zwCmd": "zwReplaceFailedNode"}
Description: Used to replace a non-responding node with a new one. The node id of the new node is the same as the node id of the replaced device. Please note that the response object properties are mostly the same as when starting an inclusion procedure since both basically do the same thing.
Response object properties:

- `type`: “INCLUSION”
- `Network`: “ZWAVE”
- `status`:
    - STARTED: Replace failed node procedure started
    - NODE_FOUND: Z-Wave module has detected a device running in inclusion mode
    - DONE: Node replaced
    - FAILED: Failed to replace node, cause of failure set in `reason` property
- `reason`: cause of the failed node replace failure

Start learn mode

Request Topic: request/networks/UUID/command

Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages

Example Request Topic Payload:

{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON", “zwDevUUID”: “UUID”, "zwCmd": "zwstartCtrlLearnMode", “nwi”:false}

Request object properties:

- `nwi`: true or false, enable or disable network wide inclusion

Description: Learn mode is used for a number of things:

- Add the controller to a network
  - Controller will receive new node id and home id
  - Controller will receive basic data about all the devices in the network
  - Controller will perform only basic configuration for each device in the network (Query each device for command classes that should be encrypted, set associations to primary controller and this controller)
  - The network role of the added controller depends on the role of the including controller:
    - If the including controller is a regular primary controller the controller included will become a secondary controller. Secondary controller can NOT add or remove other device to the network.
    - If the including controller is a primary SIS controller the controller included will become an inclusion controller. Inclusion controller can add or remove other devices from the network but it cannot independently assign new node ids to devices added to the network. Inclusion controller will ask the primary controller to assign a node id to new devices added to the network
- Exclude the controller from the network
  - Effectively, this is the same as calling hard reset. Controller will receive new home ide and node id 1. All device data will be deleted.
- Replicate data from primary controller to this controller
  - This is used to update device data if devices were added or removed by the primary controller
- Transfer primary controller role from other controller to this controller

Response object properties:

- `type`: “CTRL_LEARN_MODE”
- `Network`: “ZWAVE”
- `status`:
  - DONE: Learn mode done. Outcome of the procedure is set in the `outcome` property
  - FAILED: Learn mode failed, cause of failure set in `state` property
  - TIMEOUT: Learn mode timed out. Cause of timeout set in `state` property
- `state`: cause learn mode failure
- Outcome: can be set to the following values:

- INCLUSION
  - Controller included into network
- EXCLUSION
  - Controller excluded from network
- REPLICATION
  - Device data replicated from primary controller to this controller
- PRIMARY_ROLE_RECEIVED
  - This controller is now the primary controller

These commands are only applicable to devices that support the WAKE_UP command class. All battery powered devices (except FLIRS devices) support this command class.

Set device wakeup interval
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/devices/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON",
"value":60,"zwDevUUID":"e0a52321-88d1-4f6a-9b83-9aba6afdcf27","zwCmd":"ZwConfHandler_SET_WAKEUP"}
Description: Set device wakeup interval. Wakeup interval is defined in minutes
Response object properties:

- `type`: "SET_WAKEUP"
- `Network`: "ZWAVE"
- `Status`:
    - STARTED
    - WAITING_WAKEUP_NOTIFICATION: Waiting for the device to wake up
    - TIMEOUT_WAKEUP_NOTIFICATION: Timed out while waiting for the device to wake up (30 seconds)
    - DONE
    - FAILED: reason of failure set in `reason` property
- `value`: new wakeup interval, in minutes
- `Reason`: cause of failure

Get device wakeup interval
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/devices/UUID/messages
Example Request Topic Payload:
{"className":"com.zipato.event.CommandEvent","command":"DISCOVERY_ON",
"zwDevUUID":"e0a52321-88d1-4f6a-9b83-9aba6afdcf27","zwCmd":"ZwConfHandler_GET_WAKEUP"}
Description: Get device wakeup interval. Wakeup interval is defined in minutes
Response object properties:

- `type`: "GET_WAKEUP"
- `Network`: "ZWAVE"
- `Status`:
    - STARTED
    - WAITING_WAKEUP_NOTIFICATION: Waiting for the device to wake up
    - TIMEOUT_WAKEUP_NOTIFICATION: Timed out while waiting for the device to
    - DONE
    - FAILED: reason of failure set in `reason` property
- `value`: current wakeup interval, in minutes
- `Reason`: cause of failure

Z-Wave device firmware upgrade
Request Topic: request/networks/UUID/command
Response/Result Topic: /local/ha/bridges/zipato/networks/UUID/messages
Example Request Topic Payload:

```
{
  "className": "com.zipato.event.CommandEvent",
  "command": "DISCOVERY_ON",
  "zwCmd": "fwUpgrade",
  "zwDevUUID": "b9cc5be6-356a-468b-a78f-41e9c0a3f49a",
  "filePath": "/root/fw.hex"
}
```

Description: Starts the OTA firmware update procedure.
Response object properties:

- `type`: “ZW_DEV_OTA”
- `Network`: “ZWAVE”
- `status`:
    - STARTED: Firmware update procedure started
    - IN_PROGRESS: Firmware update progress indication, in increments of 4 percent
    - DONE: Firmware update procedure successful
    - FAILED: Failed to remove node, cause of failure set in `reason` property
- `reason`: cause of the failed node removal failure
- `devUUID`: UUID of device being upgraded
- `percent`: Firmware update progress

Device topics

Get device data
Request Topic: request/devices/UUID/info
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/devices/UUID/info
Response Type: JSON
Description: Retrieves device specific data, current state and a list of device endpoints
Payload Example:


```
{
	"link": "devices/3354b3a4-3d90-4317-aa70-5c4ba4103f38",
	"config": {
		"className": "com.zipato.network.zwave.ZwaveDevice",
		"uuid": "3354b3a4-3d90-4317-aa70-5c4ba4103f38",
		"name": "Danfoss Thermostat",
		"cv": 0,
		"sv": 0,
		"deleted": false,
		"nd": true,
		"tags": null,
		"order": null,
		"descriptorFlags": null,
		"locationId": null,
		"templateId": null,
		"description": null,
		"model": "8005-0001",
		"serial": null,
		"firmware": null,
		"periodicallyWakeUp": true,
		"status": "ENABLED",
		"user": null,
		"hidden": false,
		"wakeUpInterval": 300000,
		"savedWakeUpInterval": 300000,
		"showIcon": false,
		"iconId": 0,
		"userIconId": 0,
		"iconColors": null,
		"configuration": null,
		"zwManufacturerId": 2,
		"productTypeId": -32763,
		"productId": 1,
		"appVersion": 2,
		"appSubVersion": 6,
		"nodeId": 7,
		"type": "SLAVE",
		"alwaysListening": false,
		"minWakeUpInterval": 60000,
		"maxWakeUpInterval": 900000,
```

```
                "defaultWakeUpInterval": 300000,
                "wakeUpIntervalStep": 60000,
                "sensor1000ms": false,
                "sensor250ms": false,
                "iconType": null,
                "roleType": null,
                "basicDevClass": "ROUTING_SLAVE",
                "genericDevClass": "THERMOSTAT",
                "specificDevClass": "SETPOINT_THERMOSTAT",
                "sleeping": false,
                "eventMap": null,
                "listening": false,
                "usesStateChangeNotification": false,
                "noBatteryCheck": false,
                "assocGrpBlacklist": [],
                "crc16Encap": false,
                "neighborNodes": ["ZIPATO_CONTROLLER_NODE_ID_1"],
                "securelyIncluded": null,
                "acceptMulticasts": false,
                "alternateVnodeAssoc": false,
                "associationGroups": null,
                "manufacturerId": "2",
                "parent": "f5f264a0-0473-4c10-92c7-0fcfd1cebd5c"
        },
        "endpoints": [{
                "link": "/endpoints/7a8c09b1-aa57-476d-9a8a-5610241e0755",
                "name": "Thermostat",
                "uuid": "7a8c09b1-aa57-476d-9a8a-5610241e0755"
        }],
        "network": {
                "link": "/networks/f5f264a0-0473-4c10-92c7-0fcfd1cebd5c",
                "name": "Zwave",
                "uuid": "f5f264a0-0473-4c10-92c7-0fcfd1cebd5c"
        },
        "state": {
                "device": "3354b3a4-3d90-4317-aa70-5c4ba4103f38",
                "batteryTimestamp": 1459802516562,
                "sentTimestamp": 1459802517766,
                "online": true,
                "trouble": true,
                "receiveTimestamp": 1459802516863,
                "onlineState": "TROUBLE",
                "mainsPower": false,
                "batteryLevel": 70
        },
        "templateId": null,
        "uuid": "3354b3a4-3d90-4317-aa70-5c4ba4103f38"
}
```


Response object properties:

- `state:` Current state of the device. Please note that `state` properties are **NOT** persisted across reboots
    - `Device`: device UUID
    - `batteryTimestamp`: UNIX timestamp of last received battery status from device

- `sentTimestamp`: UNIX timestamp of last transmission from controller to device
- `Online`: online status, whether the device is reachable and in working order
- `Trouble`: Usually indicates that there is an intermittent problem with device communication. Usually it means that esither the controller failed to send a transmission to a device or a device has not reported back in some time
- `receiveTimestamp`: UNIX timestamp of last received transmission from a device
- `onlineState`: current online state
- `mainsPower`: whether a device is connected to mains or not
- `batteryLevel`: last reported battery level

Receive current device state
Request Topic: N/A
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/devices/UUID/status
Response Type: JSON
Description: Device status events are sent to this topic as device state is changed.
For instance, a device status event will be sent each time a device sends a battery status report
Payload Example:


```
"state": {
        "device": "3354b3a4-3d90-4317-aa70-5c4ba4103f38",
        "batteryTimestamp": 1459802516562,
        "sentTimestamp": 1459802517766,
        "online": true,
        "trouble": true,
        "receiveTimestamp": 1459802516863,
        "onlineState": "TROUBLE",
        "mainsPower": false,
        "batteryLevel": 70
}
```

Endpoint topics

Get endpoint data
Request Topic: request/endpoints/UUID/info
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/endpoints/UUID/info
Response Type: JSON
Description: Retrieves endpoint specific data, parent device and networkand a list of cluster endpoints
Response Example:


```
{
	"link": "endpoints/7a8c09b1-aa57-476d-9a8a-5610241e0755",
	"attributes": [],
	"clusterEndpoints": [{
		"link": "/clusterEndpoints/cb9ba097-f89c-4d20-ac10-71d42d25fde9",
		"name": "Clock",
		"uuid": "cb9ba097-f89c-4d20-ac10-71d42d25fde9"
	}, {
		"link": "/clusterEndpoints/4aa2f95e-40e6-4622-8487-0f0c866fb7e4",
		"name": "Thermostat Setpoints",
		"uuid": "4aa2f95e-40e6-4622-8487-0f0c866fb7e4"
	}],
	"config": {
		"className": "com.zipato.network.zwave.ZwaveEndpoint",
		"uuid": "7a8c09b1-aa57-476d-9a8a-5610241e0755",
		"name": "Thermostat",
		"cv": 0,
		"sv": 0,
		"deleted": false,
		"nd": true,
		"tags": null,
		"order": null,
		"type": null,
		"category": "ACTUATOR",
		"locationId": null,
		"iconType": null,
		"hidden": false,
		"status": "ENABLED",
		"description": null,
		"templateId": null,
		"descriptorFlags": null,
		"showIcon": false,
		"iconId": 0,
		"userIconId": 0,
		"iconColors": null,
		"genericDevClass": "THERMOSTAT",
		"configuration": null,
		"specificDevClass": "SETPOINT_THERMOSTAT",
		"optionalFunc": false,
		"cmdClassServer": ["COMMAND_CLASS_MULTI_CMD", "COMMAND_CLASS_CLOCK",
"COMMAND_CLASS_THERMOSTAT_SETPOINT", "COMMAND_CLASS_VERSION", "COMMAND_CLASS_PROTECTION",
```

```
"COMMAND_CLASS_MANUFACTURER_PROPRIETARY", "COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE",
"COMMAND_CLASS_BATTERY", "COMMAND_CLASS_MANUFACTURER_SPECIFIC", "COMMAND_CLASS_WAKE_UP"],
                        "cmdClassClient": ["COMMAND_CLASS_MULTI_CMD", "COMMAND_CLASS_CLOCK",
"COMMAND_CLASS_MANUFACTURER_PROPRIETARY", "COMMAND_CLASS_CLIMATE_CONTROL_SCHEDULE"],
                        "zwIconType": null,
                        "groupId": 0,
                        "mainEndpoint": true,
                        "multiInstanceId": 0,
                        "cmdClassVersionMap": {
                                "COMMAND_CLASS_WAKE_UP": 2
                        },
                        "tag": null,
                        "epId": 0,
                        "eventsToGroup": null,
                        "eventsToInstance": null,
                        "mute": false,
                        "securityCmdClasses": null,
                        "ignoreAssoc": false,
                        "parent": "3354b3a4-3d90-4317-aa70-5c4ba4103f38"
                },
                "device": {
                        "link": "/devices/3354b3a4-3d90-4317-aa70-5c4ba4103f38",
                        "name": "Danfoss Thermostat",
                        "uuid": "3354b3a4-3d90-4317-aa70-5c4ba4103f38"
                },
                "network": {
                        "link": "/networks/f5f264a0-0473-4c10-92c7-0fcfd1cebd5c",
                        "name": "Zwave",
                        "uuid": "f5f264a0-0473-4c10-92c7-0fcfd1cebd5c"
                },
                "sourceBindings": [],
                "targetBindings": [],
                "templateId": null,
                "uuid": "7a8c09b1-aa57-476d-9a8a-5610241e0755"
}
```

ClusterEndpoint topics

Get clusterEndpoint data
Request Topic: request/clusterEndpoints/UUID/info
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/clusterEndpoints/UUID/info
Response Type: JSON
Description: Retrieves cluster endpoint specific data, parent device, endpoint and network and a list of child attributes.
Response Example:


```
{
	"link": "clusterEndpoints/cb9ba097-f89c-4d20-ac10-71d42d25fde9",
	"attributes": [],
	"category": "OTHER",
	"clusterClass": "com.zipato.cluster.Clock",
	"config": {
		"className": "com.zipato.cluster.zwave.ZwaveClock",
		"uuid": "cb9ba097-f89c-4d20-ac10-71d42d25fde9",
		"name": "Clock",
		"cv": 0,
		"sv": 0,
		"deleted": false,
		"nd": true,
		"tags": null,
		"order": null,
		"queryInterval": null,
		"queryAttributes": null,
		"attributeDefs": null,
		"shouldBeQueried": true,
		"isSecurity": false,
		"hidden": false,
		"templateId": null,
		"showIcon": false,
		"iconId": 0,
		"userIconId": 0,
		"iconColors": null,
		"clusterClass": "com.zipato.cluster.Clock",
		"parent": "7a8c09b1-aa57-476d-9a8a-5610241e0755"
	},
	"device": {
		"link": "/devices/3354b3a4-3d90-4317-aa70-5c4ba4103f38",
		"name": "Danfoss Thermostat",
		"uuid": "3354b3a4-3d90-4317-aa70-5c4ba4103f38"
	},
	"endpoint": {
		"link": "/endpoints/7a8c09b1-aa57-476d-9a8a-5610241e0755",
		"name": "Thermostat",
		"uuid": "7a8c09b1-aa57-476d-9a8a-5610241e0755"
	},
	"network": {
		"link": "/networks/f5f264a0-0473-4c10-92c7-0fcfd1cebd5c",
		"name": "Zwave",
```

```
                "uuid": "f5f264a0-0473-4c10-92c7-0fcfd1cebd5c"
        },
        "templateId": null,
        "uuid": "cb9ba097-f89c-4d20-ac10-71d42d25fde9"
}
```


Attribute topics

Get attribute data
Request Topic: request/attributes/UUID/info
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/attributes/UUID/info
Response Type: JSON
Description: Retrieves attribute data.
Response example:


```
{
        "uuid": "b0334f5b-572d-4bbf-851b-356c76161ca4",
        "name": "VALVE_POSITION",
        "attribute": "value",
        "attributeId": 8,
        "clusterUuid": "bef380ce-3fb6-4e8f-8518-3563c7252787",
        "reported": true,
        "unit": "%",
        "precision": 0,
        "scale": 1.0
}
```

Attribute values
Topic: /local/ha/bridges/zipato/attributes/UUID/attributeChangeEvents
Request Topic: N/A
Request Topic Payload: N/A
Response Topic: /local/ha/bridges/zipato/attributes/UUID/info
Response Type: JSON
Description: Unsolicited attribute change events.
Response example:


```
{
        "className": "com.zipato.event.ThermostatSetpointValueEvent",
        "clusterClass": "com.zipato.cluster.ThermostatSetpoint",
        "attribute": "value1",
        "value": 21.0,
        "timestamp": 1459803817093,
        "setpointType": "HEATING",
        "source": "7a8c09b1-aa57-476d-9a8a-5610241e0755"
}
```

Attribute actions

Set attribute
Request Topic: request/attributes/UUID/textValue
Response/Result Topic: /local/ha/bridges/zipato/attributes/UUID/attributeChangeEvents
Request Topic Payload Type: String
Payload example: true, false (for binary switches for example)
Description: Set attribute state with plain strings

Set attribute
Request Topic: request/attributes/UUID/value
Response/Result Topic: /local/ha/bridges/zipato/attributes/UUID/attributeChangeEvents
Request Topic Payload Type: JSON
Payload example: (for binary switches for example)

```
{
        "value": "true"
}
```

Description: Set attribute state with simple JSON objects

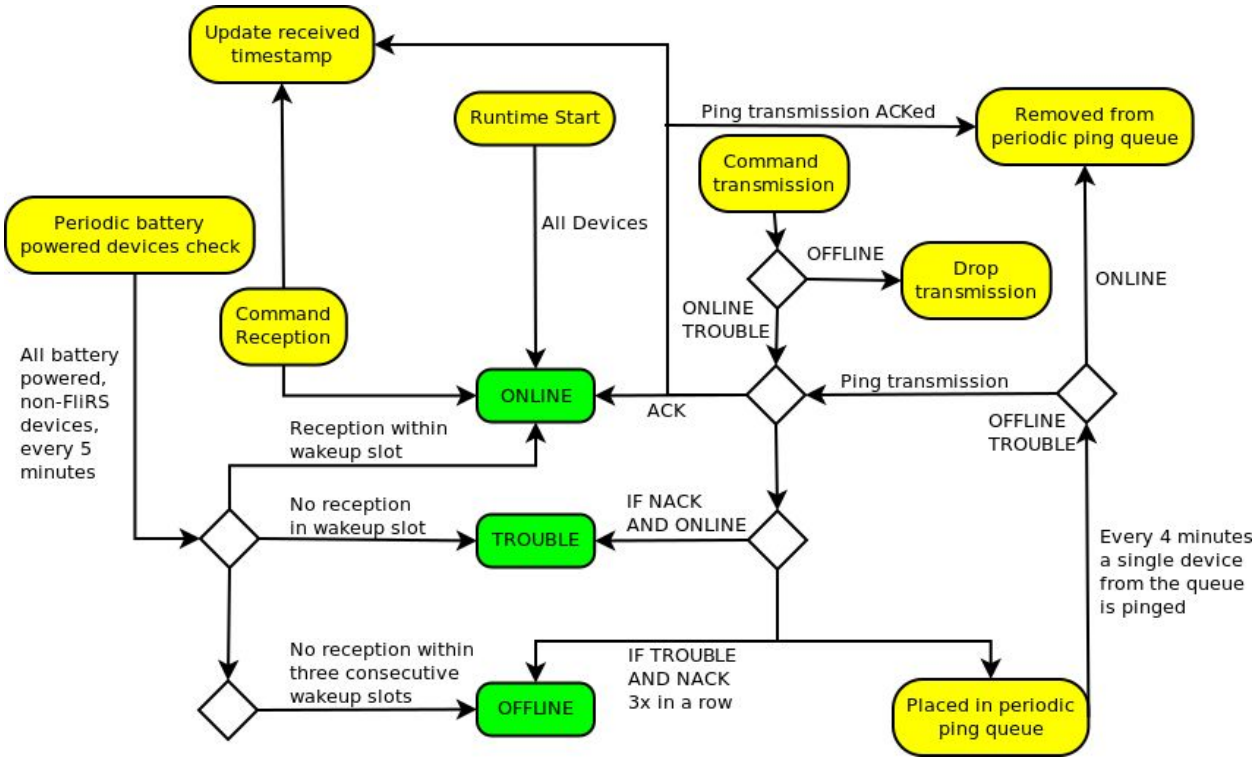
Get current attribute value
Request Topic: request/attributes/UUID/getValue
Response/Result Topic: /local/ha/bridges/zipato/attributes/UUID/currentValue
Request Topic Payload Type: N/A
Example response:

```
{"value":"No Active Events","timestamp":"2016-08-09T10:38:15Z"}
```

Description: Get current attribute value. At the moment the values are not persisted to permanent storage.

Z-Wave device state transitions

- A device can be in three states:
  - ONLINE - The controller can send and receive transmissions from the device without issues
  - TROUBLE- Intermittent transmission or reception problems
  - OFFLINE - Three consecutive transmissions have failed, or no reception from a battery powered device for three consecutive wakeup intervals, the device is considered unreachable
- State transition events are sent to `/local/ha/bridges/zipato/devices/$UUID/status` MQTT topics
- The state machine diagram consists of the following elements:
  - **Actions** - Yellow objects
  - **States** - Green objects
  - **Branches** - White objects



Binary sensor endpoints

Certain devices can have more than one binary sensor. These sensors will usually be presented as individual endpoints with a **single** (meaning such an endpoint has only one cluster) `ZwaveAlarmSensor` or `ZwaveSensorSensorBinary` cluster. Such endpoints can be distinguished with the `epId` property. A table of epId values for endpoints that contain either a `ZwaveAlarmSensor` or a `ZwaveSensorSensorBinary` is shown below.

Sensor type	`ZwaveAlarmSensor`	`ZwaveSensorSensorBinary`
ALL_PURPOSE/GENERAL_PURPOSE	156	49
SMOKE	157	50
CO	158	51
CO2	159	52
HEAT	160	53
WATER	161	54
FREEZE	162	55
TAMPER	163	56
AUX	164	57
DOOR_WINDOW	165	58
TILT	166	59
MOTION	167	60
GLASS_BREAK	168	61

Z-Wave manufacturer specific device identification tables

Please note that the numbering scheme is **arbitrary**! Each manufacturer can define their own numbering scheme.

Aeon Labs RGBW Bulb
Manufacturer ID: 0x0086

Region	EU	US	AU	CN
Product Type ID	0x0003	0x0103	0x0203	0x1D03
Product ID	0x0062	0x0062	0x0062	0x0062

Fibaro FGMS-001 Motion Sensor (Z-Wave Plus Variant)
Manufacturer ID: 0x010F

Region	EU	US	AU	RU	IL
Product Type ID	0x0801	0x0801	0x0801	0x0801	0x0801
Product ID	0x1001	0x2001	0x3001	0x4001	0x7001

Fibaro FGFS-101 Flood Sensor (Z-Wave Plus Variant)
Manufacturer ID: 0x010F

Region	EU	US	AU	RU	IL
Product Type ID	0x0B01	0x0B01	0x0B01	0x0B01	0x0B01
Product ID	0x1002	0x2002	0x3002	0x4002	0x7002

Fibaro FGFS-101 Flood Sensor
Manufacturer ID: 0x010F

Region	EU	US	AU	RU	IL
Product Type ID	0x0B00	0x0B00	0x0B00	0x0B00	0x0B00
Product ID	0x1001	0x2001	0x3001	0x4001	0x7001

Fibaro FGK-101 Door/Window Sensor (Z-Wave Plus Variant)
Manufacturer ID: 0x010F

Region	EU	US	AU	RU	IL
Product Type ID	0x0701	0x0701	0x0701	0x0701	0x0701
Product ID	0x1001	0x2001	0x3001	0x4001	0x7001

Kwikset Smartcode door lock
Manufacturer ID: 0x0090

Region	US
Product Type ID	0x0001
Product ID	0x0001

Invoke interface

Invoke interface is a simple JSON RPC mechanism which can be used to invoke methods of certain objects. The invoke interface is available for the following object types:

- `networks`
- `devices`
- `endpoints`
- `clusterEndpoints`

Request topic:
`request/OBJECT_TYPE/UUID/invoke`

Response topic:
`request/OBJECT_TYPE/UUID/messages`

Request JSON payload properties:

- `methodName` - Self explanatory, the name of the method that needs to be invoked
- `arg[1-9]` - method arguments

Request JSON payload example:

```
{
  "methodName": "userCodeSet",
  "arg1": 1,
  "arg2": "1111"
}
```

Response JSON object:

- Each invocation will return a result indicating a successful invocation, or an error. A successful invocation will always return: `"Invocation success: methodName"`
    - If an error occurs an error message will be returned. An error usually indicates an invalid method name, invalid number of arguments, wrong argument type or a JSON parsing error
- Additionally, certain methods will have specific response objects

Available methods

---

setName

Description: Change the name of the target object
Method name : `setName`
Argument count : `1`
Argument 1 : `object name`
Argument 1 type : `string`
Target object : `all objects`

---

userCodeSet

Description: Set specific door lock user code.
Method name: `userCodeSet`
Argument count: 2
Argument 1: User code slot
Argument 1 type: type: Integer
Argument 2: PIN Code
Argument 2 type: String (Please note that available characters are the ones available on the device keypad!)
Target object: `ZwaveUserCodeInput` cluster

---

userCodeGet

Description: Retrieve specific door lock user code.
Method name: `userCodeGet`
Argument count: 1
Argument 1: User code slot
Argument 1 type: Integer
Target object: `ZwaveUserCodeInput` cluster

setIgnoreDeprecatedCc

Description: Disables handling of deprecated command classes for Z-Wave Plus devices. Current list of deprecated command classes:

SENSOR_BINARY (replaced by NOTIFICATIONS command class)
ALARM_SENSOR (replaced by NOTIFICATIONS command class)

Method name: `setIgnoreDeprecatedCc`
Argument count: 1
Argument 1: true/false (false by default)
Argument 1 type: boolean
Target object: `Zwave` network

---

setAssociationPolicy

Description: set Z-Wave association policy for Z-Wave Plus devices. Current available association policies:

LIFELINE_ONLY - Only associate with Lifeline association group
ALL_GROUPS - Associate with all association groups

Method name: `setAssociationPolicy`
Argument count: 1
Argument 1: name of association policy
Argument 1 type: String
Target object: `Zwave` network

Runtime backup procedure

Since the entire runtime is completely contained within a single folder, the simplest backup and restore procedure would simply be to backup and restore the entire runtime folder. The `lib/` folder might be removed to save storage space.

Z-Wave module protocol data backup/restore mechanisms also store and read data from the zipato runtime folder.

So a complete backup procedure could go like this:

1) Backup Z-Wave module protocol data (We suggest that this is performed periodically)
2) Backup the entire Zipato runtime folder. The `lib/` folder might be removed to reduce storage requirements
3) Restore the runtime folder to a new hardware unit.
4) Copy the `lib/` folder to the runtime folder if it was previously removed from the backup
5) Restore Z-Wave module protocol data