zencontrol Docs

Jul 10, 2025

Contents

MQTT Documentation

Overview

MQTT integration is designed to allow for flexible and realtime ingestion of DALI ECD and ECG analytics. A controller is configured to point at a MQTT broker with appropriate certification to allow communication between endpoints. To ensure secure communication between endpoints, TLS v1.2 is used to provide encryption throughout data transport.

To enable the MQTT integration module for zencontrol MQTT integration, contact the zencontrol support team for licensing information.

Supported Devices

Below is a list of supported controllers for MQTT integration:

1. Application Controller Pro
2. Field Controller
3. ACx3 Pro

Setup

To setup a control system to use MQTT it's required to setup a device and change from the default parameters. To do such, a user must configure the control system settings through cloud.zencontrol.com.

Cloud

All controller settings are modified through the use of cloud.zencontrol.com.

Changing MQTT Settings

1. Navigate to cloud.zencontrol.com
2. Open your site
3. `Dashboard` -> `Gridview`
4. On the top navigation menu, click on `Add-ons`
5. On the `Add-ons` navigation menu click on `MQTT`
6. Find your controller and modify settings

Adding Keys & Certificates

1. Navigate to cloud.zencontrol.com
2. Open your site
3. `Dashboard` -> `Gridview`
4. On the top navigation menu, click on `Site`
5. On the `Site` navigation menu go to `Key store`
6. On the line that says `+` double click on the `label` cell and a label for your key/certificate
7. Double click the `Upload` cell and upload your site key(s)
    - *Note: Currently only PEM is x509 is supported*
8. You certificate/key is now uploaded to the site Key Store

Server IP & Port

Option	Default	Optional
Server IP	0.0.0.0	Required
Server Port	1883	Required

The target server IP and Port of the MQTT broker.

Note: When using TLS it's recommended to use a port other than 1883, most brokers recommend to use port 8883, however, this will depend on the deployment setup for the target MQTT broker

Username & Password

Option	Default	Optional
Username	No Default	Required
Password	No Default	Required

Usernames and passwords may be required to connect to a server and can be set through the cloud. If unused, any value will suffice to allow for connection.

Modes

Option	Default	Optional
Mode	Normal	Required

There are currently two supported operating modes for MQTT, Normal and single-endpoint. Normal operating mode splits topic up based on the data being reported, whereas Single-Endpoint targets a single topic with payload descriptors.

TLS

Option	Default	Optional
CA	Not Defined	Required
Enable SSL	True	Required

See before modifying this setting *Adding Keys & Certificates* for a guide on how to add keys to your sites key store. Any keys within the key store will be selectable from the cell dropdown.

To enable TLS, a Certificate Authority Certificate must be uploaded and configured on cloud.zencontrol.com. If an invalid Certificate is uploaded, the device will fail to authenticate and will be unable to connect until a valid certificate is uploaded.

Not Recommended: TLS Can be disabled by setting `Enable SSL = False`.

Base Topic

The base topic of the device will publish with the following definition:

```
zencontrol/{schema_version}/{serial}_{EAN}
```

Example:

```
zencontrol/v1/06571626575E_000000000007A6BB
```

Topics

When sending telemetry to the MQTT Broker in normal operation, the control system will follow a publish string standard split into 5 main sections. Single-Endpoint only publishes to the one topic. Much like the control system, the device being described is identified through the topic using their GTIN, Serial and Logical Index. This is done to ensure uniqueness on every topic, and maintain concurrency through multiple sessions where DALI Identifying information could change, such as DALI Address.

As of v2.1.54: Retention of messages is now set to 1. Clients reconnecting to the broker will receive the most recent messages for each topic.

Controller MQTT controllers support the LWT topic. If the broker detects controller disconnect, the uptime message will be published as all 0s in each of the parameters (session_id, uptime_secs, next_update_before_UTC, broker_connected_UTC).

Any controller disconnect from broker will result in controller republishing events for all topics, with the latest information.

ID Example:

```
{device_GTIN}_{device_serial}_{logical_index}
```

In use:

```
zencontrol/v1/06571626575E_000000000007A6BB/ecd/065716265220_0000000000000006_00
```

Controller

Details pertaining to the controller

```
'{base_topic}/{message_type}'
```

```
'zencontrol/v1/06571626575E_000000000007A6BB/profile'
```

Supported Messages:

- Description
- Error
- Profile
- Uptime

Group

Information regarding the status of groups

```
'{base_topic}/group/{group_id}/{message_type}'
```

```
'zencontrol/v1/06571626575E_000000000007A6BB/group/0/'
```

Supported Messages:

- Group Description
- Level
- Occupied
- Error
- Current Scene
- Light Colour

ECG

Information/details pertaining to Electronic Control Gears

```
'{base_topic}/ecg/{id}{message_type}'
```

```
'zencontrol/v1/06571626575E_000000000007A6BB/ecg/012345678910_0000000000000001_00/'
```

Supported Messages:

- Description
- Level Configuration
- Level
- Scene
- Group
- Fade
- Error
- Current Scene
- Light Colour

ECD

Information/details pertaining to Electronic Control Devices

```
'{base_topic}/ecd/{id}/{message_type}'
```

```
'zencontrol/v1/06571626575E_000000000007A6BB/ecd/012345678910_0000000000000001_00/'
```

Supported Messages:

- Description
- Lux
- Motion
- Absolute Button
- Button
- Error

Messages

- Non Specific telemetry information from the controller*

```
'{base_topic}/messages/{message_type}'
```

```
'zencontrol/v1/06571626575E_000000000007A6BB/messages/outbound'
```

Single-Endpoint

Single-Endpoint publish topic.

```
'{base_topic}'
```

```
'zencontrol/v1/06571626575E_000000000007A6BB'
```

Outbound Messages

Telemetry sent to the MQTT broker is sent using JSON. All messages will contain a session ID that changes on every initial connection to the broker. Below is an example JSON block sent to the broker.

```
{
    "session_id": 123,

    "{variable_1}": "xyz",
    "{variable_2}": "abc",

    ...
}
```

Payloads

Description

The description message type is used to describe the device currently sitting on the topic.

Payload Topic *Controller*

`{base_topic}`

ECG, ECD

`{base_topic}/ecg/{device}`

`{base_topic}/ecd/{device}`

Payload JSON

```
{
    "session_id": {session_id: int},

    "id": {device_id: string},
    "label": {device_label: string},
    "type": {device_type: int},
    "dali_address": {device_dali_address: int},
    "serial_number": [{device_serial_number: string}],
    "firmware_v_maj": {device_firmware_major_version: int},
    "firmware_v_min": {device_firmware_minor_version: int},
    "device_id": {device_id: int},
    "firmware_v_patch": {device_firmware_patch_version: int},
    "firmware_v_variant": {device_firmware_variant_version: int},

}
```

Payload Variables

Name	Type	Description
id	string	Device ID
label	string	Current Device Label
type	string	Device Type
dali_address	int	Current DALI Address of the device
serial_number	string	Hex string of serial number
firmware_v_maj	int	Device Firmware Major Version
firmware_v_min	int	Device Firmware Minor Version
device_id	int	Device ID for logical index
firmware_v_patch	int	Device Firmware Version Byte 3 (may be zero for ECG/ECD)
firmware_v_variant	int	Device Firmware Version Byte 4 (may be zero for ECG/ECD

Payload Definitions

Variable	Value	Meaning
type	0	Control Gear
type	1	Control Device

Level Configuration

Provides information to describe current min/max level settings of the device. Note that group values are based on the lowest min and the highest max (ie, works best if the targets are all the same). Last heard scene is presented but scene message gives better context.

Payload Topic

`{base_topic}/ecg/{device}/level`

`{base_topic}/group/{group}/level`

Payload JSON

```
{
    "session_id": {session_id: int},

    "max": {arc_max: int},
    "min": {arc_min: int},
    "last_heard": {last_heard_scene: int}
}
```

Payload Values

Name	Type	Description
`max`	int	Max ARC Level
`min`	int	Min ARC Level
`last_heard`	int	Last heard scene

Level

Provides information to describe the current lighting level of the device.

Payload Topic

`{base_topic}/ecg/{device}/level/value`

`{base_topic}/group/{group}/level/value`

Payload JSON

```
{
    "session_id": {session_id: int},

    "arc": {arc: int}
}
```

Payload Values

Name	Type	Description
arc	int	Current arc level

Scene

Provides information to describe the scene settings of device.

Payload Topic

```
{base_topic}/ecg/{device}/scene
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "scenes": [
        {
            "number": {scene_number},
            "arc": {scene_arc},
            "temp": {scene_temperature_in_mirek}
        },

        {
            "number": {scene_number},
            "arc": {scene_arc},
            "temp": {scene_temperature_in_mirek}
        },

        ...
    ]
}
```

Payload Values

Name	Type	Description
`scenes`	list	List of scene information
`scenes[n].number`	int	Scene Number
`scenes[n].arc`	int	Scene ARC level
`scenes[n].Temperature`	int	Scene temperature value in mirek

Current Scene

Provides information to describe current scene of the device (or group). Gives context for the last heard scene as well as whether the device is currently at the scene. For example, if you call SCENE 0, the last heard scene will be set to 0 and the at_scene will be set to 1. If the device/group is modified (by dims, colour temperatures, recall maxes etc), the last heard scene will remain 0 but the target will no longer be at_scene (and thus, at_scene will be set to 0)

Payload Topic

`{base_topic}/ecg/{device}/scene/current_scene`

`{base_topic}/group/{index}/scene/current_scene`

Payload JSON

```
{
    "session_id": {session_id: int},

    "last_heard": {last_heard_scene: int},
    "at_scene": {at_scene: int},
}
```

Payload Values

Name	Type	Description
last_heard_scene	int	Last Heard Scene
at_scene	int	1 = no adjustments to the light have occurred since scene command, 0 = scene was called but light arc or colour changed due to dim/colour/command

Group

Provides group membership information of device

Payload Topic

```
{base_topic}/ecg/{device}/group
```

Payload JSON

```
{
    "session_id": {session_id: int},
    "membership": [{group_0}, {group_1}, {group_[n]}, ... {group_15}]
}
```

Payload Values

Name	Type	Description
`membership`	list	List of group membership status
`membership[n]`	int	Current status of membership for group (0 or 1)

Fade

Information used to describe a DALI-2 Fade.

Payload Topic

```
{base_topic}/ecg/{device}/fade
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "fade_time": {fade_time: int},
    "fade_rate": {fade_rate: int},
    "number_of_steps": {number_of_steps: int},
    "dimming_curve": {dimming_curve_type: int}
}
```

Payload Variables

Name	Type	Description
fade_time	int	DALI-2 Fade Time
fade_rate	int	DALI-2 Fade Rate
number_of_steps	int	DALI-2 Fade Steps
dimming_curve	int	Current dimming curve in use

Payload Definitions

Variable	Value	Meaning
Dimming Curve	0	Linear Curve
Dimming Curve	1	Log Curve

Error

Provides information to describe an error state reported by the control module or device being reported on.

Payload Topic The payload topic for errors is slightly different to most other topics. As each error is an individual point, the error message is used as part of the topic string.

For single endpoint errors

```
{base_topic}/{type}/{device}/error/{error_message}/
```

Error Type	Topic	Description (When error_value set)	Error Type
Short Address Mask	“short_address_is_mask”	Device has MASK (0xFF) as short address	1
Received Trash	“received_trash”	Controller has received trash	2
Communication Error	“communication_error”	Device is in communication error	3
No Space	“no_space”	No space for device	4
Device Not Recognised	“device_not_recognised”	Device can’t be recognised on controller	5
Device Not Supported	“device_not_supported”	Device isn’t supported on controller	6
Battery Duration Failure	“battery_duration_failure”	Device has failed duration test	7
Battery Failure	“battery_failure”	Battery Failure Status	8
Emergency Lamp Failure	“emergency_lamp_failure”	Device has reported emergency lamp failure	9
Function Test Delay Exceeded	“function_test_delay_exceeded”	DALI test delay exceeded (couldn’t start function test)	10
Duration Test Delay Exceeded	“duration_test_delay_exceeded”	DALI test delay exceeded (couldn’t start duration test)	11
Function Test Failure	“function_test_failure”	Device failed function test	12
Duration Test Failure	“duration_test_failure”	Device failed duration test	13
Input Device Error	“input_device_error”	Generic control device input error	14
Internal Error	“internal_error”	Generic Internal error	15
ECG Failure	“ecg_failure”	Control Gear has reported failure	16
Switch Stuck	“switch_stuck”	Device is continuously reporting switch status	17
Lamp Failure	“lamp_failure”	Device is reporting lamp failure	18

examples:

```
{base_topic}/ecg/{device}/error/communication_error/
```

```
{base_topic}/ecd/{device}/error/communication_error/
```

```
{base_topic}/em/{device}/error/battery_duration_failure/
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "error_type": {error_type: int},
    "error_message": {error_message: int},
    "error_value": {error_value: int}
}
```

Payload Values

Name	Type	Description
error_type	int	Raw error type value
error_message	string	Error Message
error_value	int	Error value reported (Usually 0 for cleared or 1 for set)

Payload Definitions

Variable	Value	Meaning
`error_type`	0	No Description
`error_type`	1	Short Address Is Mask
`error_type`	2	Received Trash
`error_type`	3	Communication Error
`error_type`	4	No Space
`error_type`	5	Device Not Recognised
`error_type`	6	Device Not Supported
`error_type`	7	Battery Duration Failure
`error_type`	8	Battery Failure
`error_type`	9	Emergency Lamp Failure
`error_type`	10	Function Test Delay Exceeded
`error_type`	11	Duration Test Delay Exceeded
`error_type`	12	Function Test Failure
`error_type`	13	Duration Test Failure
`error_type`	14	Input Device Error
`error_type`	15	Application Controller Error
`error_type`	16	Control Gear Failure
`error_type`	17	Switch Stuck
`error_type`	18	Lamp Failure

Lux

Provides information to describe the current lux level of the device.

Payload Topic

```
{base_topic}/ecd/{device}/lux/value
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "value": {lux_level: int},
    "instance": {instance_number: int}
}
```

Payload Values

Name	Type	Description
value	int	Current lux level
instance	int	Instance number reported (if available)

Motion

Provides information to describe the current motion status (occupancy) the device.

Payload Topic

```
{base_topic}/ecd/{device}/motion/value
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "value": {motion_value: int},
    "instance": {instance_number: int}

}
```

Payload Values

Name	Type	Description
value	int	Current motion/occupancy status
instance	int	Instance number reported (if available)

Button

Provides information to describe the current button press state of the device.

Payload Topic

```
{base_topic}/ecd/{device}/button
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "press_type": {press_type: int},
    "instance": {instance_number: int}
}
```

Payload Values

Name	Type	Description
press_type	int	Press Type
instance	int	Instance number reported (if available)

Payload Definitions

Variable	Value	Meaning
press_type	1	Short Press
press_type	2	Long Press

Absolute Button

Provides information to describe the current button press state of the device.

Payload Topic

```
{base_topic}/ecd/{device}/absolute_button
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "abs_value": {press_type: int},
    "instance": {instance_number: int}
}
```

Payload Values

Name	Type	Description
abs_value	int	Absolute button value
instance	int	Instance number reported (if available)

Group Description

Provides information to describe a group.

Payload Topic

```
{base_topic}/group/{group_id}
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "label": {group_label: int},
    "id": {group_id: int}
}
```

Payload Values

Name	Type	Description
`label`	string	Group label
`id`	int	Group ID

Beacon Transmitter Configuration

Provides information to describe beacon configuration for device. Payloads frames are separated into iBeacon and Eddystone frames depending on the beacon type.

Version Requirement

```
>= v1.7.7
```

Payload Topic

```
{base_topic}/beacon/{device_id}/transmitter/{beacon_id}
```

iBeacon Payload

```
{
    "session_id": {session_id: int},

    "type": {beacon_type: int},
    "interval": {transmit_interval: int},
    "tx_power": {tx_power: int},
    "ibeacon_frame": {
        "proximity_uuid": {ibeacon_uuid: int list [0:15]},
        "major": {ibeacon_major: int list [0:1]},
        "minor": {ibeacon_minor: int list [0:1]}
    }
}
```

Eddystone Payload

```
{
    "session_id": {session_id: int},

    "type": {beacon_type: int},
    "interval": {transmit_interval: int},
    "tx_power": {tx_power: int},
    "eddystone_frame": {
        "namespace": {namespace: int list [0:9]},
        "instance": {instance: int list [0:5]},
    }
}
```

Payload Values

Name	Type	Description
type	int	Type of beacon
interval	int	Beacon interval in ms
tx_power	int	Transmit power level of beacon (Gain)
eddystone_frame.namespace	list	Byte list for Eddystone namespace (10 bytes)
eddystone_frame.instance	list	Byte list for Eddystone instance (6 bytes)
ibeacon_frame.proximity_uuid	list	Byte list for iBeacon proximity UUID (16 bytes)
ibeacon_frame.major	list	Byte list for iBeacon major (2 bytes)
ibeacon_frame.minor	list	Byte list for iBeacon minor (2 bytes)

Payload Definitions

Variable	Value	Meaning
`type`	0	Beacon Disabled
`type`	1	Beacon zencontrol Discoverable
`type`	2	iBeacon
`type`	3	Eddystone
`tx_power`	0	0dBm Gain
`tx_power`	1	0dBm Gain
`tx_power`	2	3dBm Gain
`tx_power`	3	3dBm Gain
`tx_power`	4	4dBm Gain
`tx_power`	5	4dBm Gain
`tx_power`	6	4dBm Gain
`tx_power`	7	4dBm Gain
`tx_power`	8	4dBm Gain

Single Endpoint

Single endpoint concatenates the payload and meta-data into a single message.

Payload Topic

```
{base_topic}
```

Payload JSON

```
{
    "cat": {category},
    "param": {parameter},
    "index": {index},
    "payload": {payload}
}
```

Payload Values

Name	Type	Description
`category`	int	Enumerated category of the message
`parameter`	int	Enumerated parameter of the message
`index`	int	ECG Address, ECD Address, Group or data index
`payload`	array	Payload of message

Category Values

Type	Value
Self	0
Description	1
Level Base	2
Level	3
Battery	4
Lamp	5
Fault	6
Fade	7
Power State	8
Lux Base	9
Lux	10
Occupied	11
Group Membership	12
Scene Data	13
Motion Base	14
Light Colour	15
Profile	16
Button	17
Absolute Button	18
Event	19
Error	20
Group Describe	21
Instance	22
Beacon Tx	23
Beacon Device	24
System Variable	25
System Variable V2	26
Uptime	27

Parameter Values

Value	Type
self	0
emergency	1
ecg	2
ecd	3
group	4
command	5
command response	6
beacon	7

Example

```
{
    "cat": 3, // ECD
    "param": 11, // Occupied
    "index": 2, // ECD Address - 2
    "payload":
    {
        "session_id": 16, // Current Session
        "occupied": 1, // Occupancy detected
        "instance": 0 // On the 0th instance
    }
}
```

Uptime

Provides information about the uptime of the system. Also serves as LWT (last will and testament). If all parameters are set to 0, the controller is not connected to the broker.

Payload Topic

```
{base_topic}/uptime
```

Payload JSON

```
{
    "session_id": {session_id: int},

    "uptime_secs": {uptime_secs: int},
    "next_update_before_UTC": {next_update_before_UTC: int},
    "broker_connected_UTC": {broker_connected_UTC: int}
}
```

Payload Values

Name	Type	Description
uptime_secs	int	Number of seconds the controller has been up for
next_update_before_UTC	int	UTC time that an update to this topic will occur before
broker_connected_UTC	int	UTC time of connection to broker

System Variable

Provides state of system variables

Payload Topic

```
{base_topic}/system_variable/index
```

E.g /system_variable/0, /system/variable/147

Payload JSON

```
{
    "session_id": {session_id: int},

    "value": {value:int},
    "magnitude": {magnitude:int},
    "label": {label: list}
}
```

Payload Values

Name	Type	Description
value	int	Value (INT32)
magnitude	int	exponential scaling (10^x) (INT8)
label	string	Name of the system variable

Light Colour

Provides information to describe current min / max colour temperatures for a device. Note that group values are based on the lowest min and the highest max (ie, works best if the targets are all the same)

Payload Topic

`{base_topic}/ecg/{device}/light_colour`

`{base_topic}/group/{device}/light_colour`

Payload JSON

```
{
    "session_id": {session_id: int},

    "min_colour_mirek": {min_colour_mirek: int},
    "max_colour_mirek": {max_colour_mirek: int}
}
```

Payload Values

Name	Type	Description
`min_colour_mirek`	int	Min Colour Temperature in Mirek
`max_colour_mirek`	int	Max Colour Temperature in Mirek

Light Colour Value

Provides information to describe current level colour temp / RGBWAF value of the device

Payload Topic

`{base_topic}/ecg/{device}/light_colour/value`

`{base_topic}/group/{device}/light_colour/value`

Payload JSON

```
{
    "session_id": {session_id: int},

    "colour_temp_mirek": {colour_temp_mirek: int},
    "RGBWAF": {
        r: {red: int},
        g: {green: int},
        b: {blue: int},
        w: {white: int},
        a: {amber: int},
        f: {freecolour: int}
    }
}
```

Payload Values

Name	Type	Description
`colour_temp_mirek`	int	Current Colour Temperature in Mirek
`red`	int	Current Red Colour (0-254 valid, 255 not supported or known)
`green`	int	Current Green Colour (0-254 valid, 255 not supported or known)
`blue`	int	Current Blue Colour (0-254 valid, 255 not supported or known)
`white`	int	Current White Colour (0-254 valid, 255 not supported or known)
`amber`	int	Current Amber Colour (0-254 valid, 255 not supported or known)
`freecolour`	int	Current Freecolour Colour (0-254 valid, 255 not supported or known)

Devicebound Messages

Telemetry can be sent to the control module publishing to the devicebound topic. On Successful ingestion of a command, a command response will be published with a response for the given commands. Some commands have optional values, that can be omitted, furthermore, sanitisation of min and max values will be applied to certain values.

Devicebound Topic:

```
{base_topic}/messages/devicebound
```

Devicebound Payloads

Each message is categorised by the target, type, ID and request ID and may contain multiple messages to the same target. The request ID is a random value sent to the device to identify messages sent, the same request ID is sent on a message response.

Example Payload:

```
{
    "id": {target_device:int},
    "rid": {request_id:int},
    "target": {target_type:int},

    "{command_1}": {command_payload:JSON Object},
    "{command_n}": {command_payload:JSON Object},
    ...
}
```

For ECD and ECG commands, the "id" refers to the DALI Address of the device. For Beacon commands, the "id" refers to the device_id of the target beacon as beacons can be shared across multiple logical indexes.

Payload Definitions:

Variable	Value	Meaning
target	0	Control Module
target	1	Unused
target	2	ECG
target	3	ECG Group
target	4	ECD
target	5	ECD Group
target	6	Beacon

Level

Changes the current ARC and/or Temperature or RGBWAF level values of device(s). Values committed are ignored in the command, for example, if temperature isn't set, no temperature change is made.

As of 2.1.54, a mirek format (kelvin / 1000000) has been added so that the system can function entirely in mirek format, which is the native unit for dali (requests for colour temperature in kelvin may not be honoured precisely due to conversions to mirek).

The level command conforms to DALI-2 Standards, therefore, if a temperature and arc change with a fade is set, only the arc level is faded.

RGBWAF fades tend to produce less smooth transitions. For smoother colour transitions, scenes are recommended instead.

Each device's individual DALI fade time setting still applies to any fade operations. Most devices default to fade times of 0.7 or 1 second. If you request a fade duration shorter than the device's default, each new fade command will restart the fade sequence for the full default duration (0.7 or 1 second).

Color temperature changes (whether in kelvin or mirek) do not support custom fade times but the device will apply its built-in DALI fade time when executing colour temperature adjustments.

Payload JSON

```
{
    "level": {
        "arc": {arc_level:int},
        "fade_time": {fade_time_ms:int},
        "temp": {temperature_kelvin:int},
        "mirek": {temperature_mirek:int},
        "r": {red:int},
        "g": {green:int},
        "b": {blue:int},
        "w": {white:int},
        "a": {amber:int},
        "f": {freecolour:int}
    }
}
```

Payload Values

Name	Type	Description	Required	Min/Max
arc	int	Target ARC level	False	0/255
fade_time	int	Target fade time in milliseconds	False	1000/65000
temp	int	Target temperature in kelvin	False	1-65000
mirek	int	Target temperature in mirek	False	1-65000
r	int	Target Red Colour	False	1-254
g	int	Target Green Colour	False	1-254
b	int	Target Blue Colour	False	1-254
w	int	Target White Colour	False	1-254
a	int	Target Amber Colour	False	1-254
f	int	Target Freecolour Colour	False	1-254

Beacon Config

Changes beacon configuration for a device. Type is automatically changed when the appropriate inner frame is sent.

Version Requirement

```
>= v1.7.7
```

iBeacon Payload JSON

```
{
    "beacon_config": {
        "position": {beacon_position: int},
        "interval": {interval: int},
        "tx_power": {tx_power: int},
        "ibeacon": {
            "proximity_uuid": {ibeacon_uuid: int list [0:15]},
            "major": {ibeacon_major: int list [0:1]},
            "minor": {ibeacon_minor: int list [0:1]}
        }
    }
}
```

Eddystone Payload JSON

```
{
    "beacon_config": {
        "position": {beacon_position: int},
        "interval": {interval: int},
        "tx_power": {tx_power: int},
        "eddystone": {
            "namespace": {namespace: int list [0:9]},
            "instance": {instance: int list [0:5]},
        }
    }
}
```

Payload Values

Name	Type	Description	Required	Min/Max
`position`	int	Beacon index (Beacon ID)	True	0/Number of beacons supported on device
`interval`	int	Transmit interval in ms	True	0/Max transmit time time for beacon type
`tx_power`	int	Transmit power (gain)	True	0/8
`eddystone.namespace`	list	Byte list for Eddystone namespace (10 bytes)	True (if Eddystone)	0/255 per index
`eddystone.instance`	list	Byte list for Eddystone instance (6 bytes)	True (if Eddystone)	0/255 per index
`ibeacon.proximity_uuid`	list	Byte list for iBeacon proxmity UUID (16 bytes)	True (if iBeacon)	0/255 per index
`ibeacon.major`	list	Byte list for iBeacon major (2 bytes)	True (if iBeacon)	0/255 per index
`ibeacon.minor`	list	Byte list for iBeacon minor (2 bytes)	True (if iBeacon)	0/255 per index

Payload Definitions

Variable	Value	Meaning
type	0	Beacon Disabled
type	1	Beacon zencontrol Discoverable
type	2	iBeacon
type	3	Eddystone
tx_power	0	0dBm Gain
tx_power	1	0dBm Gain
tx_power	2	3dBm Gain
tx_power	3	3dBm Gain
tx_power	4	4dBm Gain
tx_power	5	4dBm Gain
tx_power	6	4dBm Gain
tx_power	7	4dBm Gain
tx_power	8	4dBm Gain

System Variable

Changes a system variable value. Magnitude is used for exponential scaling (powers of 10). Magnitude would generally be set to 0 unless interacting with the results for general purpose sensors.

Payload JSON

```
{
    "id": {system_variable_index:int},
    "rid": 0,
    "target": 0,
    "system_variable": {
        "value": {value:int},
        "magnitude": {magnitude:int},
    }
}
```

Payload Values

Name	Type	Description	Required	Min/Max
system_variable_index	int	system variable index	true	0-147
value	int	value to set	true	INT32MIN/INT32MAX
magnitude	int	exponential scaling	true	INT8MIN/INT8MAX

Example Set system variable 100 to -2010 x 10^-1

```
{
    "id": 100,
    "rid":0,
    "target": 0,
    "system_variable": {
        "value": -2010,
        "magnitude": -1
    }
}
```

Scene

Sends a scene change on a dali target

Payload JSON

```
{
    "goto_scene": {
        "target_scene": {scene_index:int},
    }
}
```

Payload Values

Name	Type	Description	Required	Min/Max
target_scene	int	Scene number	False	0-15