List of fields in the payload currently implemented:

```
[
  {
    "target": 'device', --> which type of sensor/device you wish to write to
    "cmd": 'set pow=on',   --> the command which should be applied to the target
    "cmdreg": 'as',    --> the name of the target's register where the command should be written to
    "ackreg": 'pow',    --> the name of the target's register where the command written should be acknowledged
    "ackval": 'on'    --> the value of the target's acknowledgment register where the command written should be
                          acknowledged using this value
  }
]
```

The following explanations concern commands for physical ZBS-devices.

Field "target":
--> field is optional (if omitted, value is tied to 'device')
--> possible values: 'device' (the command should be written to a physical ZBS device)
'virtual' (the command should be written to a virtual device)
'gateway' (the command should be written to the gateway)
--> currently, targeting physical device only is fully implemented, still sending commands to virtual devices or gateway is prepared only
--> value of field is not case sensitive

Field "cmd":
--> field is mandatory
--> possible values: all kind of valid commands for register of target (e.g. 'set pow=on' for 'AS'-register of ZBS-110)
--> value of field can/may be case sensitive

Field "cmdreg":
--> field is optional (if omitted, value will be written to register AS of ZBS-device)
--> possible values: all valid names of registers a ZBS- device can own (e.g. AS, SC, POW etc.)
--> the command provided in field 'cmd' will be written to this register
--> value of field is not case sensitive

Field "ackreg":
--> field is optional
--> possible values: all valid names of registers a ZBS- device can own (e.g. AS, SC, POW etc.)
--> the command written will be acknowledged using the value of the this register
--> value of field is not case sensitive

Field "ackval":
--> field is optional
--> possible values: polymorph value of all kind of 'ackreg'
--> the command written will be acknowledged comparing the value of the ack-register with this value
--> value of field is not case sensitive

How commands sent via MQTT will be acknowledged:

Generally there are no guarantees that any command will reach any ZBS device via radio (ZigBee), because messages can get lost via radio connections.
However, we implemented some techniques to acknowledge commands, which will work perfectly in usual situations where the radio networks works without severe connection problems.

In any case, if the MQTT client (gateway) receives a valid command-payload, the receipt of this command will be acknowledged by publishing an according event to the event-topic of the appropriate device. Therefore, to receive

any acknowledgements it is necessary to configure a command topic (topic where commands should be received) and an event topic (topic where events should be published).

For example, you want to switch on power at a ZBS-110 device with name FG-110-8. The minimum payload which should be published to 'cloud/GCC-GW/exec_action/FG-110-8' will look like the following:

```
[
  {
    "cmd": 'set pow=on'
  }
]
```

The following event will be published to 'events/GCC-GW/FG-110-8'

```
[
  {
    "output": "command 'set pow=on' received @ device 'FG-110-8'",
    "state": "received",
    "timestamp": "2015-01-05T13:21:19Z+0000",
    "sensor_id": "FG-110-8_AS"
  }
]
```

The 'sensor_id' of this event has following format: "sensorname"_"registername", where the name of the register is the register where the command should be written to.

So if the cloud receives this event, it is guaranteed that the command which should be written to a ZBS devices has reached the gateway, which not means that the command has been processed/send to the device by the gateway's ZigBee-Handler nor that it has been acknowledged by the target device itself.

This is the "weakest" kind of command acknowledgement.

A more expressive acknowledgement can be achieved, if the field "ackreg" is provided in the command payload, e.g.:

```
[
  {
    "cmd": 'set pow=on'
    "ackreg": 'pow'
  }
]
```

As before, the following event will be published to 'events/GCC-GW/FG-110-8' at first:

```
[
  {
    "output": "command 'set pow=on' received @ device 'FG-110-8'",
    "state": "received",
    "timestamp": "2015-01-05T13:21:19Z+0000",
    "sensor_id": "FG-110-8_AS"
  }
]
```

Furthermore, the MQTT client tries to acknowledge the command by comparing the previous value of ackreg with the new value of ackreg for a maximum time of 30 seconds. If the client detects that the value of the ackreg has changed within this time, the following event will be published:

```
[
  {
    "output": "command 'set pow=on' acknowledged @ device 'FG-110-8'",
    "state": "acknowledged",
    "timestamp": "2015-01-05T12:58:17Z+0000",
    "sensor_id": "FG-110-8_POW"
```

```
  }
]
```

The 'sensor_id' of this event has following format: "sensorname"_"ack-registername", where the name of the register is the register where the command should be acknowledged.
This is a better type of acknowledgment, because you are able to verify that the command written results in a state change. This kind of acknowledgement is still weak, because if you send the same command twice, a state change of the ack register's value is not guaranteed.

A more expressive acknowledgement can be achieved, if the field "ackreg" and the field "ackval" are provided in the command payload, e.g.:

```
[
  {
    "cmd": 'set pow=on'
    "ackreg": 'pow'
    "ackval": 'on'
  }
]
```

As before, the following event will be published to 'events/GCC-GW/FG-110-8' at first:

```
[
  {
    "output": "command 'set pow=on' received @ device 'FG-110-8'",
    "state": "received",
    "timestamp": "2015-01-05T13:21:19Z+0000",
    "sensor_id": "FG-110-8_AS"
  }
]
```

Furthermore, the MQTT client tries to acknowledge the command by comparing the value of ackreg with the value of field "ackval" for a maximum time of 30 seconds. If the client verifies (within 30 seconds) that this values are equal, the following ack-event will be published:

```
[
  {
    "output": "command 'set pow=on' acknowledged @ device 'FG-110-8'",
    "state": "acknowledged",
    "timestamp": "2015-01-05T12:58:17Z+0000",
    "sensor_id": "FG-110-8_POW"
  }
]
```

This is the strongest type of acknowledgment, because you are able to verify that the command written leads to the result which you've expected.

All kinds of acknowledgement will run in to a timeout if they exceed the time of 30 seconds (an appropriate log message will be written to the gateway's log file).

Short summary :
Required payload for ack-type "received at gateway":
[{"cmd": 'set pow=on'}] # get 'cmd received'-ackmsg only

Required payload for ack-type "received at gateway" and "value of target device register has changed":
[{"cmd": 'set pow=on', "ackreg": 'as'}] # get 'cmd received' and 'cmd acknowledged' (ack validated by change of the value of ackreg)

Required payload for ack-type "received at gateway" and "value of target device register equals expected value":

[{"cmd": 'set pow=on', "ackreg": 'pow', "ackval": 'on'}] # get 'cmd received' and 'cmd acknowledged' (ack validated by verifying that value of ackreg equals ackval)

The field "cmdreg" has been introduced, because you are free to not only write commands to the AS-register, you can also write to other registers of the ZigBee device directly.
For example the following command payloads result in the same action (switch power on @ ZBS-110), with slightly different content of the event messages:

Command payload (variant 1):

```
[
  {
    "cmd": 'set pow=on'
    "ackreg": 'pow'
    "ackval": 'on'
  }
]
```

Receive-event payload (for variant 1):

```
[
  {
    "output": "command 'set pow=on' received @ device 'FG-110-8'",
    "state": "received",
    "timestamp": "2015-01-05T13:21:19Z+0000",
    "sensor_id": "FG-110-8_AS"
  }
]
```

Ack-event payload (for variant 1):

```
[
  {
    "output": "command 'set pow=on' acknowledged @ device 'FG-110-8'",
    "state": "acknowledged",
    "timestamp": "2015-01-05T12:58:17Z+0000",
    "sensor_id": "FG-110-8_POW"
  }
]
```

Command payload (variant 2):

```
[
  {
    "cmd": 'on'
    "cmdreg": 'pow'
    "ackreg": 'pow'
    "ackval": 'on'
  }
]
```

Receive-event payload (for variant 2):

```
[
  {
    "output": "command 'on' received @ device 'FG-110-8'",
    "state": "received",
    "timestamp": "2015-01-05T13:21:19Z+0000",
    "sensor_id": "FG-110-8_POW"
  }
]
```

Ack-event payload (for variant 2):


```
[
  {
    "output": "command 'on' acknowledged @ device 'FG-110-8'",
    "state": "acknowledged",
    "timestamp": "2015-01-05T12:58:17Z+0000",
    "sensor_id": "FG-110-8_POW"
  }
]
```