Send commands to the BOS

You can use the Mongo database to apply changes to the BOS.

- Introduction
- Change a point value
- Create a schedule regular event
- Delete a schedule regular event
- Create a schedule special event
- Update a schedule special event
- Delete a schedule special event
- Ack an alarm

Introduction

There is a dedicated collection called "messages". Unless the other collections where the BOS creates the documents per asset, you, as a third party, will create documents in this collection.

One document is dedicated to one change request you want to apply to the BOS. The BOS is listening in real-time to every document you may create.

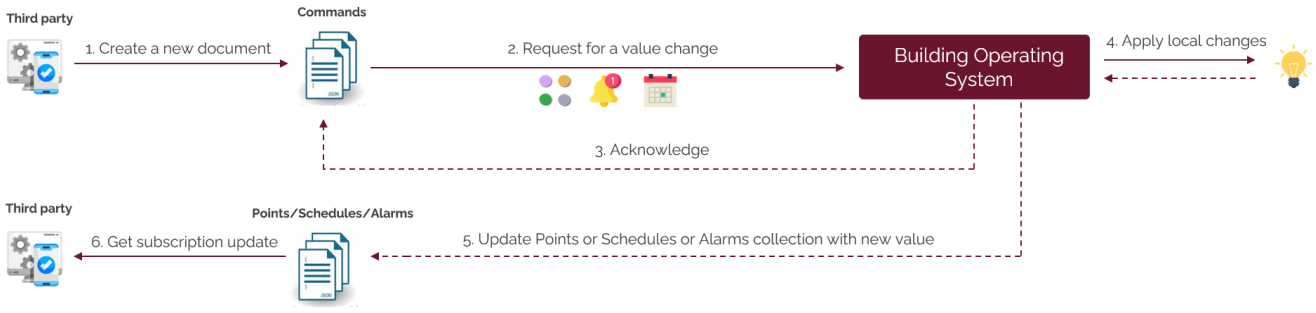
Once the document is created you can no longer modify it, it's pointless, only new documents are handled by the BOS

Once the BOS learnt about a new document (== a demand for a change), the BOS will edit the document and add a new field "ack" with the current timestamp. This is meant to tell you "ok, got it" but it doesn't mean the change was applied correctly because there may be several communication layers to pass through to deliver your change request.

Only when the change is effectively applied and the BOS get the feedback value, it will update the associated document (in the points, schedules or alarms collection) so you can subscribe to it to receive the final value.

It would be possible for example that you request a change to a certain value, you receive a slightly different value. Example, on a control damper, there is usually a min opening value embedded in the controller. If you ask for 0%, the controller may allow only a minimum of 50%, so you would get "50" after a change request of "0".

The diagram below shows the sequence order:



Each type of of change request will require a specific document syntax as shown after.

Change a point value

Let's focus now on a Point data type value change.

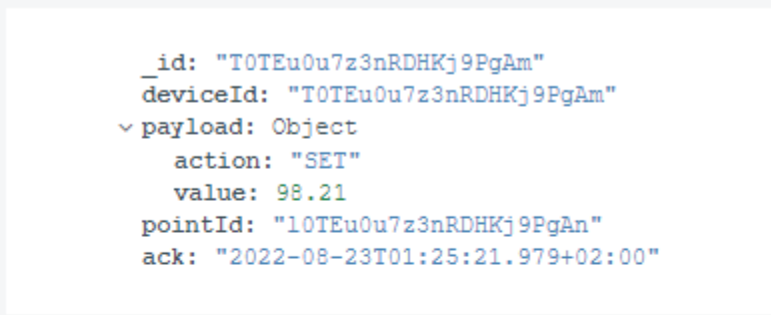
This is the following format to create for the document to create to request a point value change:

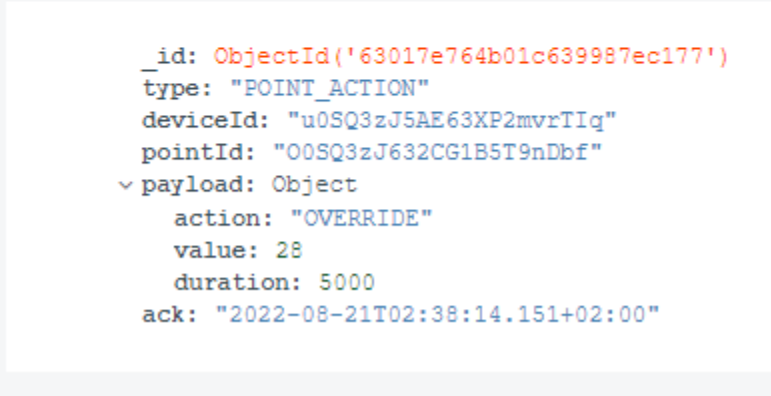
Example

```
{
    "type": "POINT_ACTION",
    "deviceId": "<DeviceId>",
    "pointId": "<PointId>",
    "payload": {
        "action": "<action>",
        "value": <value to be set>,
        "duration": <override duration>
    }
 }
```

- **deviceId**: this is read on the Point Data asset. It represents the container such as a Device asset, Equipment asset, Model asset. The deviceId value will always match one of these assets. (There is no link at all with the concept of "device")
- **pointId**: unique point identifier of the Point Data asset
- **action**: they are listed here: Point data type (SET/OVERRIDE/EMERGENCY_OVERRIDE/AUTO/EMERCY_AUTO
- **value**: the type of value will depend on the primary value of the point. You can read it from the tag "controlType"
- **duration**: the duration will only be used with the OVERRIDE action, otherwise it's useless.

Two examples below. The ack field is added by the BOS.





A request for change could be denied by the BOS in the following conditions:

- The point is not "writable", it's a read-only point.
- The point has been set not to be changed by a third party. Ask the MSI in this case, indicating your pointId.

Create a schedule regular event

It's possible to create regular events on a Schedule. See Schedule data type for more information about regular events.

The document syntax to do so is as follow:

Example

```
{
  "type": "CREATE_RECURRING_EVENT",
  "deviceId": "<DeviceId>",
  "scheduleId": "<ScheduleId>",
  "payload": {
    "eventId": "<Your own id>",
    "weekday": "<weekday>",
    "startTime": "<startTime>",
    "endTime": "<endTime>",
    "value": <value>
  }
}
```

- **deviceId**: this is read on the Schedule Data asset. It represents the container such as a Device asset, Equipment asset, Model asset. The deviceId value will always match one of these assets. (There is no link at all with the concept of "device")
- **scheduleId**: unique schedule identifier of the Schedule Data asset
- **eventId**: Your own unique identifier for the event. It's useful to persist it on your side, if you want to delete it later or simply to read it among the other regular events.
- **weekday**: The target weekday (example: monday)
- **startTime** and **endTime**: ISO 8601 time to start the regular event. It's recommended to use UTC.
- **value**: The type of value will depend on the primary value of the point. You can read it from the tag "controlType"

Delete a schedule regular event

It's possible to delete a regular event on a Schedule if you know its unique id (the event id). See Schedule data type for more information about regular events.

The document syntax to do so is as follow:

Example

```
{
  "type": "DELETE_RECURRING_EVENT",
  "deviceId": "<DeviceId>",
  "scheduleId": "<ScheduleId>",
  "eventId": "<Your own id>"
}
```

- **deviceId**: this is read on the Schedule Data asset. It represents the container such as a Device asset, Equipment asset, Model asset. The deviceId value will always match one of these assets. (There is no link at all with the concept of "device")
- **scheduleId**: unique schedule identifier of the Schedule Data asset
- **eventId**: Your own unique identifier for the event.

Create a schedule special event

It's possible to create special events on a Schedule. See Schedule data type for more information about special events.

The document syntax to do so is as follow:

Example

```
{
  "type": "CREATE_SPECIAL_EVENT",
  "deviceId": "<DeviceId>",
  "scheduleId": "<ScheduleId>",
  "payload": {
    "eventId": "<Your own id>",
    "eventName": "<Your own event name>",
    "startDate": "<startDate>",
    "endDate": "<endDate>",
    "value": <value>
  }
}
```

- **deviceId**: this is read on the Schedule Data asset. It represents the container such as a Device asset, Equipment asset, Model asset. The deviceId value will always match one of these assets. (There is no link at all with the concept of "device")
- **scheduleId**: unique schedule identifier of the Schedule Data asset
- **eventId**: Your own unique identifier for the event. It's useful to persist it on your side, if you want to delete it later or simply to read it among the other special events.
- **eventName**: A name that the user will see in the BOS UI or through an app as the Supervision app.
- **startDate** and **endDate**: ISO 8601 date (not time) to start and end the special event. It's recommended to use UTC.
- **value**: The type of value will depend on the primary value of the point. You can read it from the tag "controlType"

Update a schedule special event

It's possible to update a special event if you know its unique id that was used to create it.

The document syntax to do so is as follow:

Example

```
{
  "type": "UPDATE_SPECIAL_EVENT",
  "deviceId": "<DeviceId>",
  "scheduleId": "<ScheduleId>",
  "eventId": "<Your own id>",
    "payload": {
      "startDate": "<startDate>",
      "endDate": "<endDate>",
      "value": <value>
  }
}
```

- **deviceId**: this is read on the Schedule Data asset. It represents the container such as a Device asset, Equipment asset, Model asset. The deviceId value will always match one of these assets. (There is no link at all with the concept of "device")
- **scheduleId**: unique schedule identifier of the Schedule Data asset
- **eventId**: Your own unique identifier for the event. It's useful to persist it on your side, if you want to delete it later or simply to read it among the other special events.
- **startDate** and **endDate**: ISO 8601 date (not time) to start and end the special event. It's recommended to use UTC.
- **value**: The type of value will depend on the primary value of the point. You can read it from the tag "controlType"

Delete a schedule special event

It's possible to delete a special event on a Schedule if you know its unique id that was used to create it. See Schedule data type for more information about special events.

The document syntax to do so is as follow:

Example

```
{
  "type": "DELETE_SPECIAL_EVENT",
  "deviceId": "<DeviceId>",
  "scheduleId": "<ScheduleId>",
  "eventId": "<Your own id>"
}
```

- **deviceId**: this is read on the Schedule Data asset. It represents the container such as a Device asset, Equipment asset, Model asset. The deviceId value will always match one of these assets. (There is no link at all with the concept of "device")
- **scheduleId**: unique schedule identifier of the Schedule Data asset
- **eventId**: Your own unique identifier for the event. It's useful to persist it on your side, if you want to delete it later or simply to read it among the other special events.

Ack an alarm

To acknowledge an alarm, you will need to retrieve its "uuid", this is the unique identifier of an alarm event.

Example

```
{
    "type": "ACK_ALARM",
    "payload": {
        "uuid": "<uuid of the alarm to ack>"
    }
 }
```