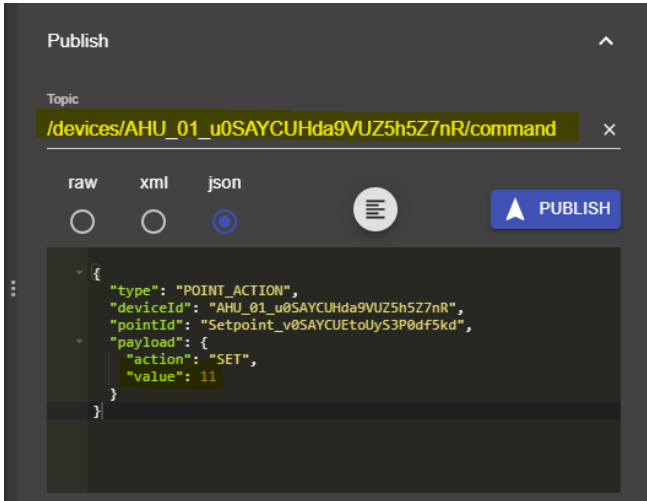# Step 5 Send commands from mqtt to Niagara

MQTT Connector supports broker to niagara communication via mqtt messages to control points remotely.

Send a point action command

1. Go to the Mqtt Explorer application then choose the topic of the device you want to send commands to.

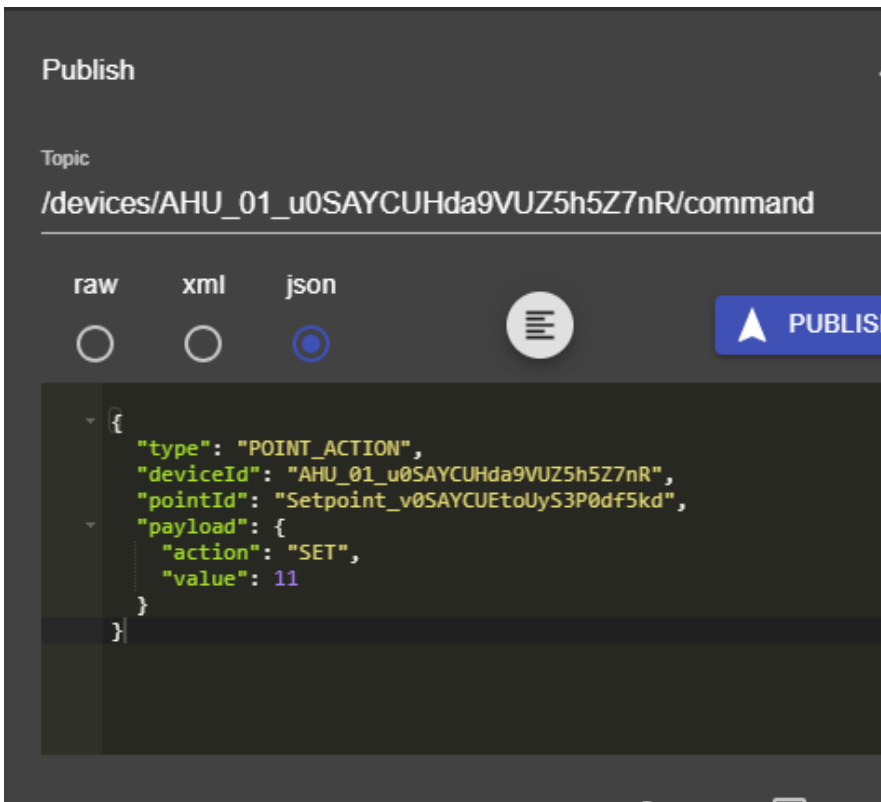   You can change the command topic name in the connector configuration.

2. By default we use this message template for **POINT_ACTION** command. but you can use any format that meets your needs. check the connector advanced settings.
   a. This is the default command template.

```
{
        "type": "POINT_ACTION",
        "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
        "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
        "payload": {
                "action": "SET",
                "value": 11
        }
}
```
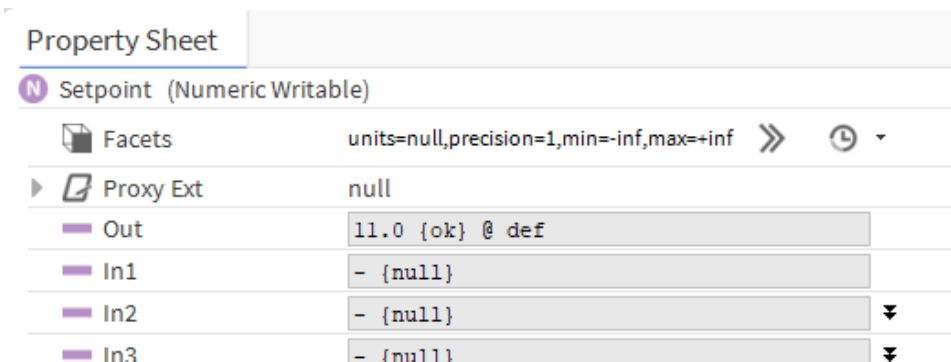
   b. This is the default connector point action command configuration.

3. Hit publish.

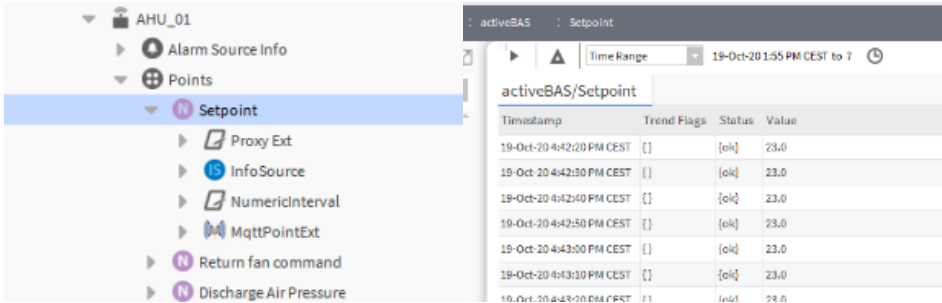4. On the niagara you should see the point value applied successfully.

5. And the new value sent to the broker.

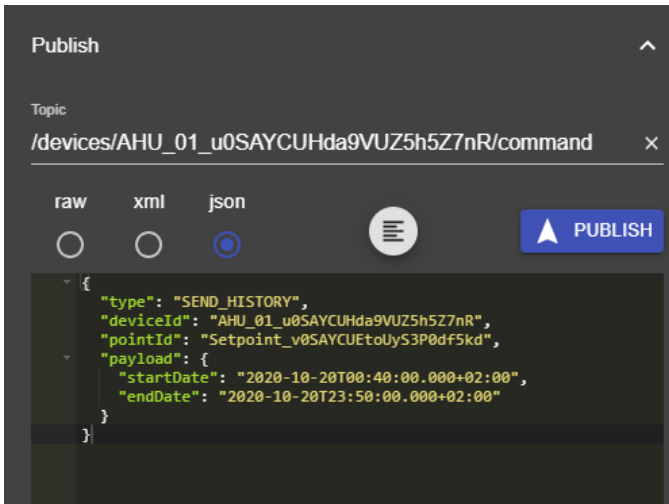
Send a send history command

You can pull historical value for any point that has a history associated by sending a **SEND_HISTORY** command.

1. Add a history. extension to the point.

2. Go to the mqtt application and send the command to the device.

3. By default we use this message template for send history command. but you can use any format that meets your needs. check the connector advanced settings.

    a. This is the default command template.

```
{
        "type": "SEND_HISTORY",
        "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
        "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
        "payload": {
                "startDate": "2020-10-20T00:40:00.000+02:00",
                "endDate": "2020-10-20T23:50:00.000+02:00"
        }
}
```

    b. This is the default connector send history command configuration.

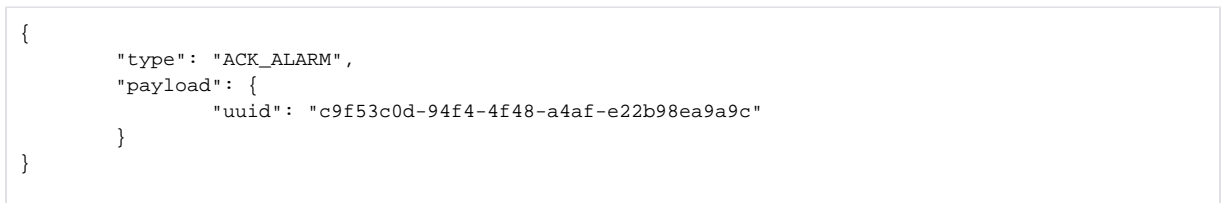|	
Commands Policy	Single Point Command
Message Type	{json('type')}
Command Set Object	{json('').escape}
Command Device Id	{json('deviceId')}
Command Point Id	{json('pointId')}
Command Action	{json('payload.action')}
Command Value	{json('payload.value')}
Command Duration	{json('payload.duration')}
Start Date	{json('payload.startDate')}
End Date	{json('payload.endDate')}
Delta	{json('payload.delta')}
Roll Up	{json('payload.rollup')}

4. And you should see the messages being sent.
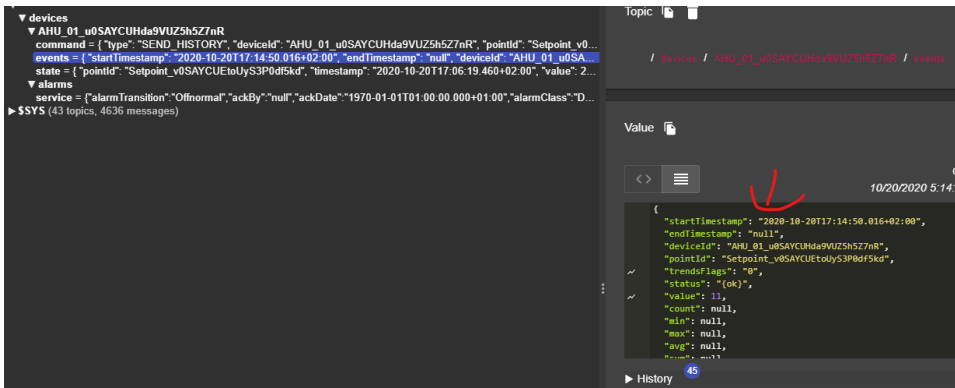
To change the message format check the connector advanced setting then history message template

![icon] **History Message Variables**	$(startTimestamp) $(endTimestamp) $(pointId) $(deviceId) $(status) $(value) $(trendsFlags) $(count) $(min) $(max) $(avg) $(sum)

![icon] **History Message Template**	```
	{
	    "startTimestamp": "$(startTimestamp)",
	    "endTimestamp": "$(endTimestamp)",
	    "deviceId": "$(deviceId)",
	    "pointId": "$(pointId)",
	    "trendsFlags": "$(trendsFlags)",
	    "status": "$(status)",
	    "value" : $(value),
	    "count": $(count),
	    "min": $(min),
	``

Send ack alarm command

You can ack alarms by sending an **ACK_ALARM** command to any alarm recipient device.

1. By default we use this message template for ack alarm command. but you can use any format that meets your needs. check the connector advanced settings.
   a. This is the default command template.

```
{
        "type": "ACK_ALARM",
        "payload": {
                "uuid": "c9f53c0d-94f4-4f48-a4af-e22b98ea9a9c"
        }
}
```

   b. This is the default connector send history command configuration.

	"min": $(min),
![icon] Alarm Uuid	{json('payload.uuid')}
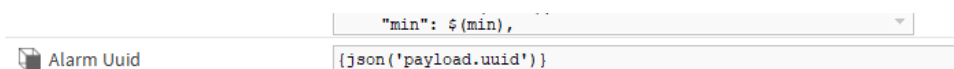
2. Go the alarms console and pick an unacked alarm id.

Timestamp	Source State	Ack State	Source	Alarm Class	Priority	Message Text
🔔 20-Oct-20 9:32:49 AM CEST	Normal	Acked	Active Power A	HVAC_Level1	255	To Normal Power P
🔔 20-Oct-20 9:35:21 AM CEST	Normal	Acked	Active Power A	HVAC_Level1	255	To Normal Power P
🔔 20-Oct-20 9:43:19 AM CEST	Normal	Unacked	Active Power A	HVAC_Level1	255	To Normal Power P
🔔 20-Oct-20 9:53:50 AM CEST	Normal	Unacked	Active Power A	HVAC_Level1	255	To Normal Power P
🔔 20-Oct-20 9:59:19 AI						
🔔 20-Oct-20 10:01:51 /						
🔔 20-Oct-20 10:07:19 /						
🔔 20-Oct-20 10:12:49 /						
🔔 20-Oct-20 10:15:21 /						
🔔 20-Oct-20 10:27:19 /						

Alarm Record

Timestamp	20-Oct-20 9:43:19 AM CEST
Uuid	c9f53c0d-94f4-4f48-a4af-e22b98ea9a9c
Source State	Normal
Ack State	Unacked
Ack Required	true
Source	Active Power A

3. Go to the alarm device and send the command.

Publish

Topic

/devices/alarms/command

raw   xml   json

⚪   ⚪   🔘   PU

```
{
    "type": "ACK_ALARM",
    "payload": {
        "uuid": "c9f53c0d-94f4-4f48-a4af-e22b98ea9a9c"
    }
}
```
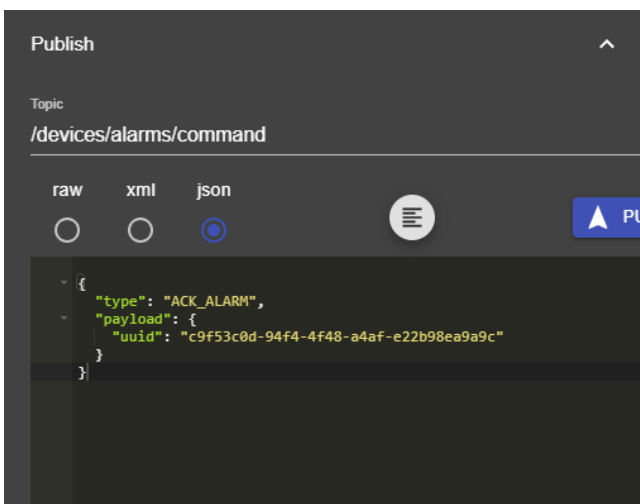
4. Go back to the console and you should see that the alarm has been acked.

Alarm History

Timestamp	Source State	Ack State	Source	Alarm Class	Priority	
🔔 20-Oct-20 9:32:49 AM CEST	Normal	Acked	Active Power A	HVAC_Level1	255	T
🔔 20-Oct-20 9:35:21 AM CEST	Normal	Acked	Active Power A	HVAC_Level1	255	T
🔔 20-Oct-20 9:43:19 AM CEST	Normal	Acked	Active Power A	HVAC_Level1	255	T
🔔 20-Oct-20 9:53:50 AM CEST	Normal	Unacked	Active Power A	HVAC_Level1	255	T
🔔 20-Oct-20 9:						
🔔 20-Oct-20 10						
🔔 20-Oct-20 10						
🔔 20-Oct-20 10						
🔔 20-Oct-20 10						
🔔 20-Oct-20 10						
🔔 20-Oct-20 10						

Alarm Record

Timestamp	20-Oct-20 9:43:19 AM CEST
Uuid	c9f53c0d-94f4-4f48-a4af-e22b98ea9a9c
Source State	Normal
Ack State	Acked
Ack Required	false
Source	Active Power A
	local:|station:|slot:/Drivers/ModbusTcpNetwork/