

# Step 5 Send commands from MongoDB to Niagara

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



You can only send commands from a cluster which is replica set, otherwise the application director will display error messages similar to this one  
com.mongodb.MongoCommandException: Command failed with error 40573 (Location40573): 'The \$changeStream stage is only supported on replica sets' on server localhost:27017. The full response is {"ok": 0.0, "errmsg": "The \$changeStream stage is only supported on replica sets", "code": 40573, "codeName": "Location40573"}

## Send a point action command

1. Go to the Mongo Compass application then add a new document to the commands collection.

Insert to Collection test.commands

VIEW

```
1 = {  
2   "type": "POINT_ACTION",  
3   "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",  
4   "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",  
5   "payload": {  
6     "action": "SET",  
7     "value": 11  
8   }  
9 }  
10 }
```

CANCEL

INSERT

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

Database Name	test
Devices Collection	devices
Points Collection	points
Alarms Collection	alarms
External Messages Collection	commands

2. By default we use this message template for **POINT\_ACTION** command. by you can use any format that meat 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.

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')}

### 3. Hit Insert.


Insert to Collection test.commands

VIEW  

```

1 {
2   "type": "POINT_ACTION",
3   "deviceId": "ANU_01_u05AYOUHde8VU2N627e8",
4   "pointId": "Setpoint_VESAYCUTUy53P9dF5k8",
5   "payload": {
6     "action": "SET",
7     "value": 11
8   }
9 }

```













CANCEL INSERT

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

#### Property Sheet

##### N Setpoint (Numeric Writable)

	Facets	units=null,precision=1,min=-inf,max=+inf	 
	Proxy Ext	null	
	Out	11.0 {ok} @ def	
	In1	- {null}	
	In2	- {null}	
	In3	- {null}	

5. And the new value sent to the broker.

test.points Documents

DOCUMENTS 54 TOTAL SIZE 27.1KB AVG. SIZE 514B INDEXES 1

Documents Aggregations Schema Explain Plan

FILTER {'tags.name': 'Setpoint'} OPTIONS FIND

ADD DATA VIEW

Displaying documents 1 - 1 of 1

```
{
  "_id": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
  "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
  "tags": {
    "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
    "status": "{ok} @ def",
    "timestamp": "2020-10-21T14:44:47.304+02:00",
    "value": 11
  }
}
```

## 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.

activeBAS : Setpoint

Time Range 19-Oct-20 1:55 PM CEST to ?

Timestamp	Trend Flags	Status	Value
19-Oct-20 4:42:20 PM CEST	[ ]	{ok}	23.0
19-Oct-20 4:42:30 PM CEST	[ ]	{ok}	23.0
19-Oct-20 4:42:40 PM CEST	[ ]	{ok}	23.0
19-Oct-20 4:42:50 PM CEST	[ ]	{ok}	23.0
19-Oct-20 4:43:00 PM CEST	[ ]	{ok}	23.0
19-Oct-20 4:43:10 PM CEST	[ ]	{ok}	23.0
19-Oct-20 4:43:20 PM CEST	[ ]	{ok}	23.0

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

Insert to Collection test.commands

VIEW

```
{
  "type": "SEND_HISTORY",
  "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
  "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
  "startdate": "2020-10-21T14:44:47.304+02:00",
  "enddate": "2020-10-21T14:50:00.000+02:00"
}
```

CANCEL INSERT

3. By default we use this message template for send history command. by you can use any format that meat 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": "2020-10-21T00:40:00.000+02:00",
      "endDate": "2020-10-21T23:50:00.000+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.

test.points

Documents Aggregations Schema Explain Plan

FILTER

ADD DATA

```

_id: "Setpoint_v0SAYCUEtoUyS3P0df5kd"
deviceId: "AHU_01_u0SAYCUHda9VUZ5h5Z7nR"
> tags: Object
pointId: "Setpoint_v0SAYCUEtoUyS3P0df5kd"
status: "{ok}"
timestamp: "2020-10-21T14:44:47.304+02:00"
value: 11
avg: null
count: null
endTimeStamp: "null"
max: null
min: null
startTimestamp: "2020-10-21T14:54:40.066+02:00"
sum: null
trendsFlags: "0"

_id: "mongoDBTest_I0SCcz9xFmic2OwCctmw"
pointId: "mongoDBTest_I0SCcz9xFmic2OwCctmw"
status: "{ok} @ def"
timestamp: "2020-10-19T10:36:25.192+02:00"

```

To change the message format check the connector advanced setting then history message template

	<code>\$(startTimestamp)</code>
	<code>\$(endTimestamp)</code>
	<code>\$(pointId)</code>
	<code>\$(deviceId)</code>
	<code>\$(status)</code>
	<code>\$(value)</code>
	<code>\$(trendsFlags)</code>
	<code>\$(count)</code>
	<code>\$(min)</code>
	<code>\$(max)</code>
	<code>\$(avg)</code>
	<code>\$(sum)</code>
History Message Variables	
History Message Template	<pre>{   "startTimestamp": "\${startTimestamp}",   "endTimestamp": "\${endTimestamp}",   "deviceId": "\${deviceId}",   "pointId": "\${pointId}",   "trendsFlags": "\${trendsFlags}",   "status": "\${status}",   "value": \${value},   "count": \${count},   "min": \${min}. }</pre>

## Send ack alarm command

You can ack alarms by sending an **ACK\_ALARM** command to any alarm recipient device.

- By default we use this message template for ack alarm command. by you can use any format that meat your needs. check the connector advanced settings.
  - This is the default command template.

```
{
  "type": "ACK_ALARM",
  "deviceId": "alarms"
  "payload": {
    "uuid": "0898a6b9-1c9c-4128-bd53-1001b261c706"
  }
}
```

- This is the default connector send history command configuration.

	<code>"min": \${min},</code>
Alarm Uuid	<code>{json('payload.uuid')}</code>

- Go the alarms console and pick an unacked alarm id.

Alarm History

Timestamp	Source State	Ack State	Source	Alarm Class	Priority	Message
21-Oct-20 11:12:50 AM CEST	Normal	Unacked	Active Power A	HVAC_Level1	255	To Normal
21-Oct-20 11:18:18 AM CEST	Normal	Unacked	Active Power A	HVAC_Level1	255	To Normal
21-Oct-20 11:20:00 AM CEST	Alarm Record					
21-Oct-20 11:28:00 AM CEST						
21-Oct-20 11:34:00 AM CEST	<b>Timestamp</b> 21-Oct-20 11:12:50 AM CEST					
21-Oct-20 11:36:00 AM CEST	<b>Uuid</b> 0898a6b9-1c9c-4128-bd53-1001b261c706					
21-Oct-20 11:36:00 AM CEST	<b>Source State</b> Normal					
21-Oct-20 11:39:00 AM CEST	<b>Ack State</b> Unacked					
21-Oct-20 11:39:00 AM CEST	<b>Ack Required</b> true					

- Go to the alarm device and send the command.









Insert to Collection test.commands

VIEW  

```
1 {
2   "type": "ACK_ALARM",
3   "deviceId": "alarms",
4   "payload": {
5     "uuid": "0898a6b9-1c9c-4128-bd53-1001b261c706"
6   }
7 }
8
```

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
 21-Oct-20 11:12:50 AM CEST	Normal	Acked	Active Power A	HVAC_Level1	255
 21-Oct-20 11:18:18 AM CEST	Normal	Unacked	Active Power A	HVAC_Level1	255
 21-Oct-20	 Alarm Record				
 21-Oct-20					
 21-Oct-20					
 21-Oct-20					
 21-Oct-20					
Timestamp		21-Oct-20 11:12:50 AM CEST			
Uuid		0898a6b9-1c9c-4128-bd53-1001b261c706			
Source State		Normal			
Ack State		Acked			
Ack Required		false			