Interact with workflows

Document version 0.1.1

Overview

This document describes different approaches to communicating with a workflow. There are different ways depending on what type of communication and the requirements of such. The following text will guide you through examples inspired by needs and conditions from cases with IOT-devices.

OBS!

This document is an updated version of the original “Communicating with MobileResponse Workflows” that was written during summer of 2017.

In our efforts to reduce complexity and make development easier, we have also decided to drop the old name of the platform/system **MobileResponse** in favor of **Bosbec we**. The reason is to make it easier for customers to know that it is one and the same company supporting the entire platform.

What this means is that documentation will be updated with new domain-names as soon as all services are migrated, tested and fully supported under the **Bosbec.io** domain name. It also means that in this document we still use the old domain names **mobileresponse.com**, that will keep working in parallel with **Bosbec.io.**

Please go to https://help.bosbec.io/InteractingWithBosbec.pdf **to make sure you are reading the most recent version of this document!**

Tutorials

“Bosbec we” is a system that constantly evolve and expand its capabilities. An example of this is the API and endpoints that **Triggers** can react to. And at the time of writing this (autumn 2017) the most recent addition to this area is the HTTP-IN endpoint.

In this section, we present 4 different approaches to communicating with workflows and in the end of this document there are example requests/responses as reference.

The traditional "Bosbec we" API (api2.bosbec.io)

The first available web-API created for "Bosbec we" originated from the need for a graphical interface (earlier versions of admin.mobileresponse.io). That first API has been replaced with a second version api2.bosbec.io.

This API can be used to create/read/update/delete most of the objects in your "Bosbec we" account. In many ways, this is the most capable API, where this focuses on both the ability to manipulate objects as well as executing workflows (with ExecutingWorkflowTriggers).

How to get started

First thing you need to use this API is a "Authentication-Token", and there are a number of different ways to obtain such a token.

A token from API2

Token via api2.bosbec.io/authenticate; You can **POST** a HTTP-request to the API with user credentials and from the response retrieve your token. This token will have a life-time of 20 minutes, and this life-time is updated each time the token is used with the API up to a limit of max 24 hours since it was issued.

To get a token you should send a POST request and use header and body from example below:

https://api2.bosbec.io/authenticate

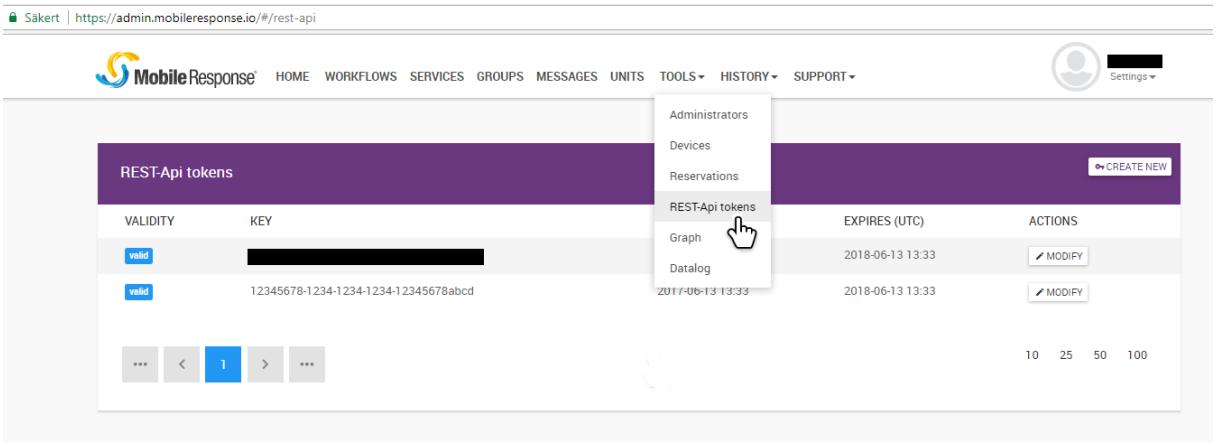
```
Content-Type: application/json

{
    "data": {
        "username": "YOUR_USERNAME_HERE",
        "password": " YOUR_PASSWORD_HERE"
}
```

When you have this token (which is the same as what admin-GUI does behind the scenes) you can use the id from the token object in the response data to access different parts of MobileResponse APIs.

A token from Admin user interface

Tokens can also be created in the graphical user interface (GUI) at admin.bosbec.io. Head over to the tools menu and select alternative REST-Api tokens. The fact that it is called REST, is in fact a 'rest' (*pun intended*) from where we could only generate these unique tokens for the API located at rest. bosbec.io. In the future, the name of these tokens will likely change to Bosbec We-Tokens.



You should now be presented with a view like the image above, and if it is your first visit then the listing might be empty.

Create a new token by clicking the “CREATE NEW” button in the upper right corner of the purple header area. There will be a short pop-up message asking if you want to generate a new token, accept and wait until the token has been created and loaded into the listing.

The difference between this token and the one you could get from the api2 (*above*) is that this token will automatically have a valid-time/expiration date one year from now. And you can change this on your own by clicking the “MODIFY” button.

Make calls to api2 with the token

The token you’ve created in either one of the steps above can now be used to give you access to call functions in api2. When using api2 communicate with a workflow you need to call the execute-with-parameters function in the workflows-area.

Below you can find example of how a request to execute-with-parameters function would look like.
In this example, we have used the **WorkflowId** (*found in admin under workflows and in details for the workflow you want to execute*).
We have also added the **TriggerNames** that can be used to only execute one (or more) ExecutingWorkflowTriggers in the workflow. You simply write the name of the triggers you want to execute (*comma separated, like this: “trigger_1, trigger_2”*). If you do not set what triggers you want to execute, then the workflow will execute all ExecutingWorkflowTriggers when this function is called.
Finally we also provide “parameters” into the execution of the workflow. What that means is that we set some **MetaData** keys and values in our request, and when the workflow executes these metadata from our request will be added or overwrite what is already set in the workflow before executing the first job.

```
POST /workflows/execute-with-parameters HTTP/1.1

Host: api2.bosbec.io
Content-Type: application/json

{
    "data": {
        "WorkflowId": "525f61e3-b8dd-4a15-b59e-320ac665a767",
        "TriggerNames": "trigger_1",
        "MetaData": {
            "my-data-1": "This is input data to the workflow"
        }
    },
    "AuthenticationToken": "1337aa6f-5e0b-43fc-b10d-144f486af632"
}
```

Requests and responses in API2

Remember the following:

- You need to provide the authentication-token-id in your request for all requests but the */authenticate* request.
- The request should be JSON-data and the header *Content-Type: application/json* should be set.
- All data in the request should be within the *“data”* object but *“authenticationToken”* should be outside the data object.
- Responses are JSON-data and the actual data that is the result of the request is inside the *“data”* object.

- Responses will provide a "*status"* parameter which most of the time will say *Success* even in cases when you get a response saying that something went. What that status indicates is that the server could accept and process the request without any unexpected behavior or crashes.

api2. bosbec.io is recommended to direct the requests to, but most of what is in the documentation found at the older api.mobileresponse.se will work with API2 as well.

Hint!

- Use debugging tools (in web-browsers or for example Fiddler if you are windows user) and monitor requests made by admin-GUI if you want to make use of the same features.

Using MQTT for two-way communication (mqtt.bosbec.io:1883)

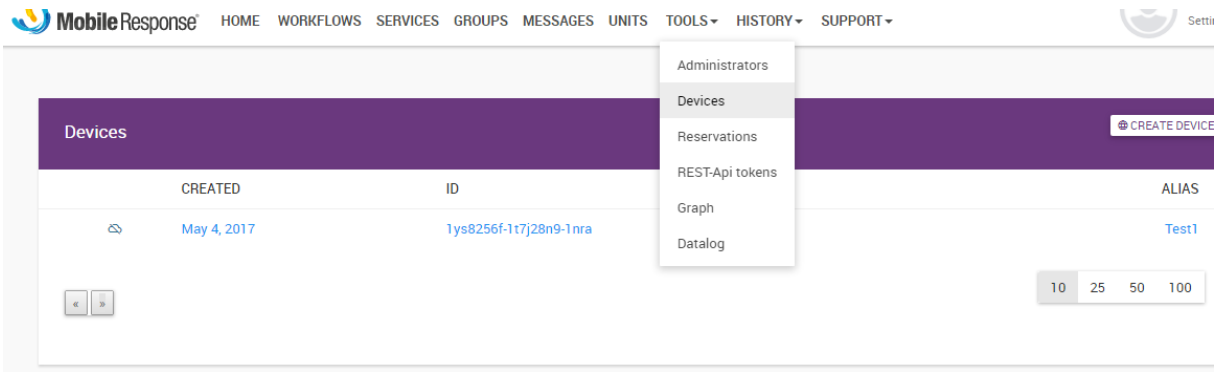
Another way to communicate with Bosbec We and workflows is by using our MQTT-broker. This way you can both listen to messages on a topic as well as post messages that may cause the IotTrigger to execute.

Before we can communicate via MQTT we need two things registered. We need a unit if we want to send a message to a MQTT-receiver from workflows and we need to register a device.

Register a device

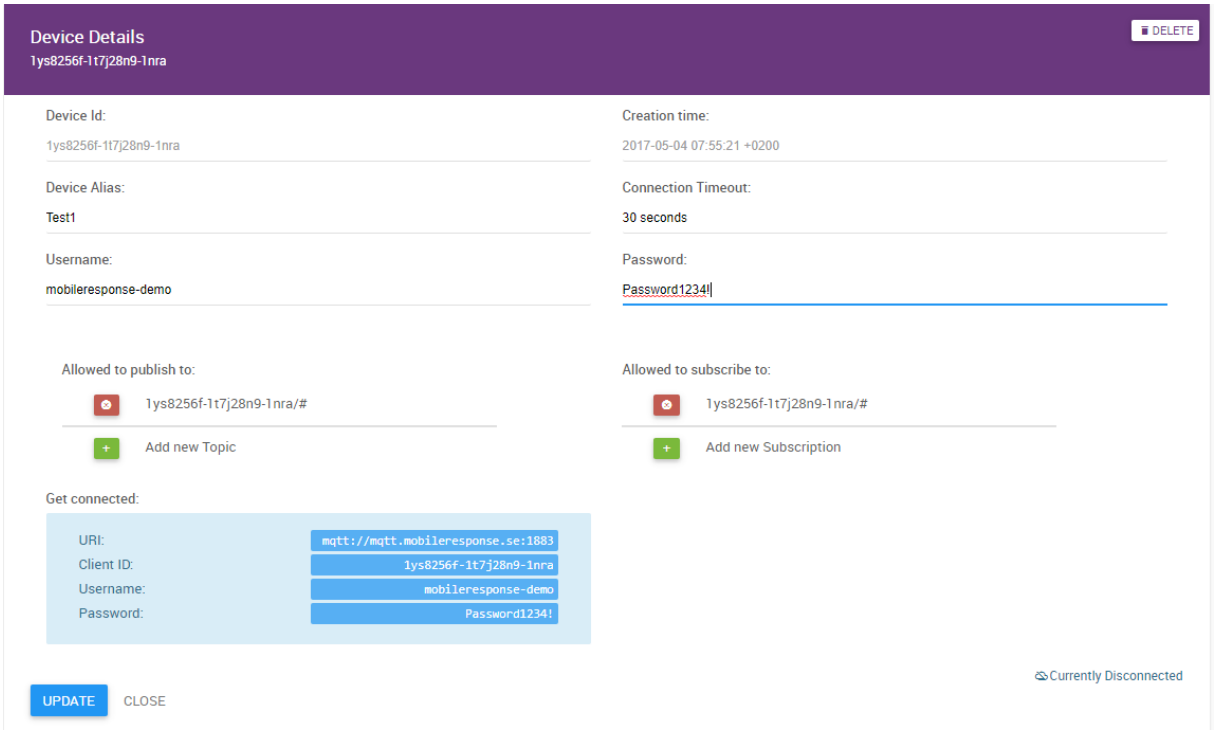
The device is first and foremost a registration of the first part of the topic, similar to a keyword that we register for Email or Sms triggers.

But the device is more than just a registration of a keyword, it is also the object that allow you to manage your credentials for connections to the MQTT-broker.



Step1. Go to admin and click your way to devices (in the tools menu)

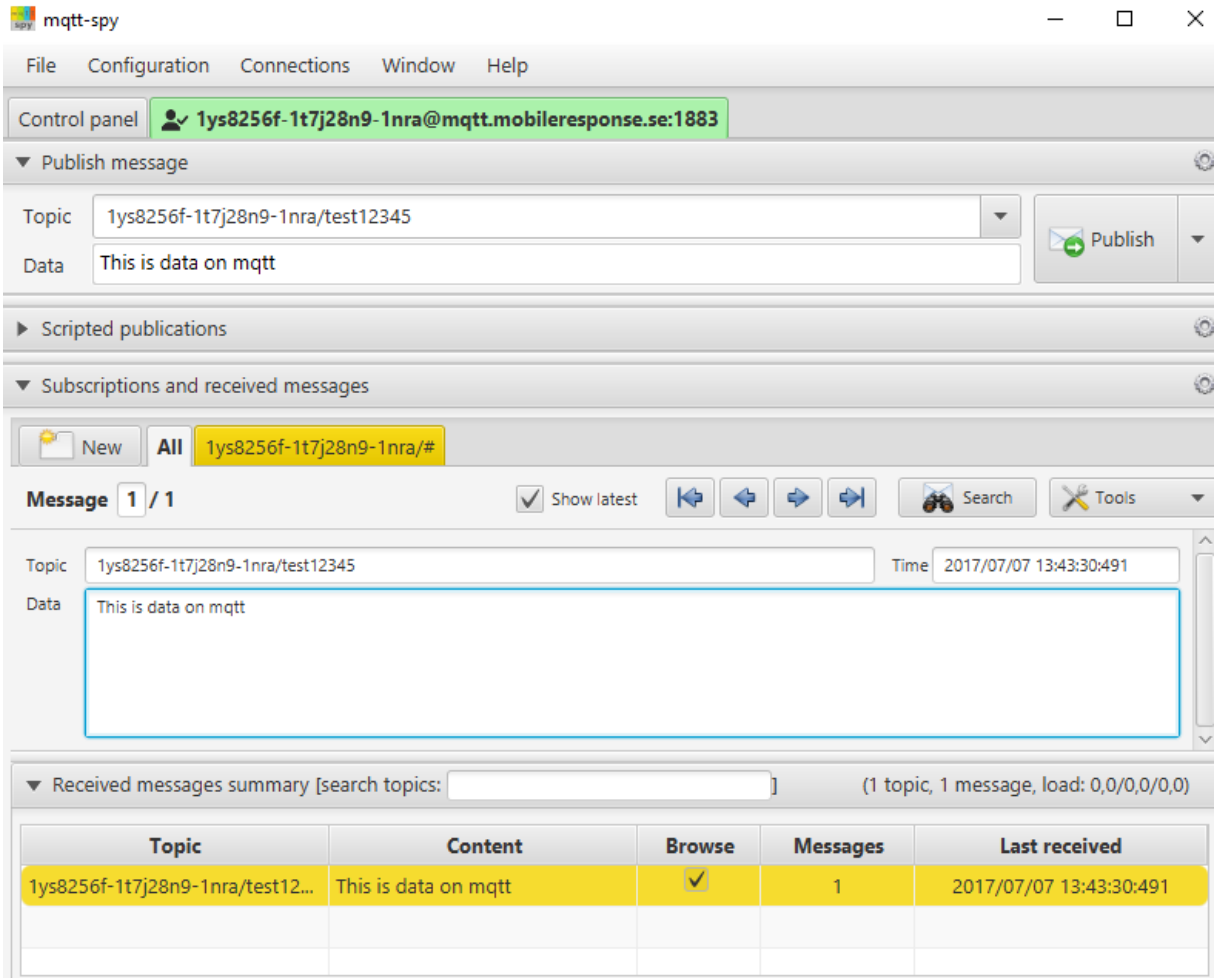
You will be presented with a listing of your current devices. If there are no devices on your account then you could go ahead and create one by clicking the “CREATE DEVICE” button.
When creating a device, the page will reload and you could see your device in the listing once it has been created.



Step2. Configure the device (after clicking your device in the listing you will be presented with the configuration).

In the example above we've set up a device (Test1) with a **username** and a **password** that we can use to connect to the mqtt-broker. The information in the "Get connected"-box gives you all information you need to connect.
We have also configured what topics we are allowed to publish or subscribe to, and you can see that we've used the **#** character that represents a "wild-card".

One way to test out your connection is to use a tool that can connect and test out your device. An example is the mqtt-spy (a java application). Try to connect and reload the Devices-page in admin and you should see that the status change, in the listing there is a green circle and in the detailed view the message "Currently Disconnected" has changed to "Currently connected".

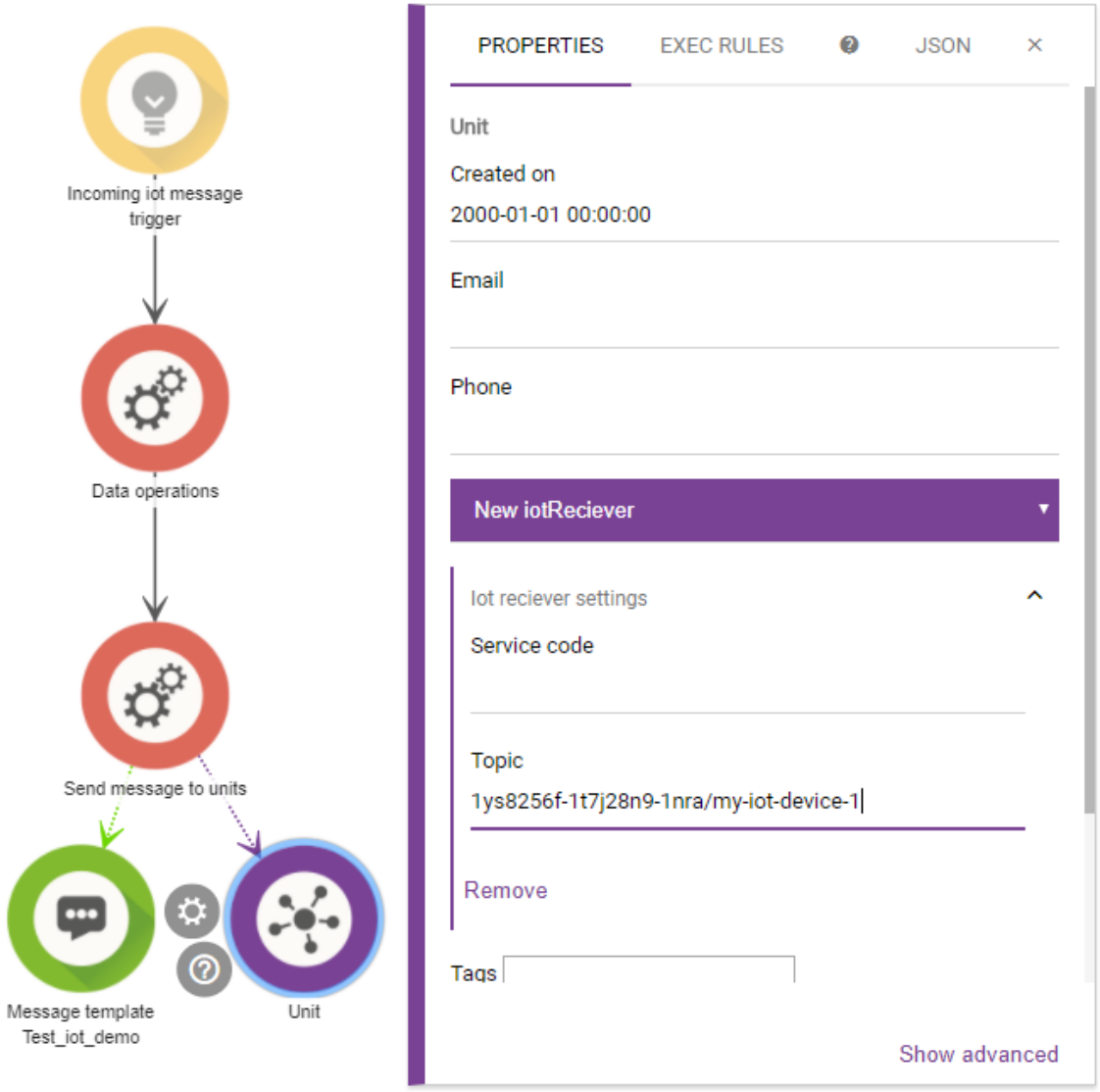


Step3. mqtt-spy application where we've connected, subscribed, sent and received a message via mqtt.bosbec.io.

Now we are done with the registration of a device, and we can continue with IOT-receiver and a workflow.

Creating unit with IOT-receiver

We need units to send messages to in MobileResponse workflows. In this case we need the unit to communicate not via Email or Phone, but over MQTT. And for that we need to set up a unit with an **IOT-receiver** and this can be done in the workflow builder.



Step1. The example above shows a workflow that can be used to test out communication over MQTT.

We drag a new **Unit** to the workflow and click it to see the details. In this case we are only interested in the **IotReceiver** fields, and more precise it is the topic that you need to set. Get the first part of your topic from the page where you've configured your device and add a forward-slash / followed by a topic that you would like to post to when communicating with that unit. In the image above we've added */my-iot-device-1* and the rest of the workflow is a DataOperations-job that generates a metadata with the current time (*metadada.time)*, and this metadata is then included in the message template with the syntax [time].

Step2. The trigger is configured with the devices topic + /test1

When we activate the workflow above we have a trigger that is ready to start a workflow as soon as a message that matches this trigger comes in from the mqtt-broker.
And the result of the workflow is that another message is sent back on the */my-iot-device-1* -topic.

REST-inspired API (rest.bosbec.io)

Bosbec We REST-Api is inspired by ideas that can be found on and around the internet under the term REST API. The key concepts for the API is that using the HTTP-verbs should mean something, URLs that provides clarity to what a request does with a resource and make use of response codes to indicate status of the request.

An example of what is mentioned above could be that you may GET the details for the workflow 824169ab-0fae-476c-b5df-b65dcd60b19c if you use HTTP-method GET and the URL
https://rest.bosbec.io/1/workflows/824169ab-0fae-476c-b5df-b65dcd60b19c

The /1/ in the URL represents the API-version or the version of that method. The /workflows/ part represent what resource we are dealing with and the id in this GET-request is the identifier so that the system may find and GET that resource for you.

POST-ing a request to execute a workflow

To start a workflow via the REST-Api you need to send a POST request to
https://rest.bosbec.io/1/workflows

There is one extra interesting thing about the rest.bosbec.io -API, and that is the ability to execute (most) workflows directly during the request and get a response configured as you specify in the request. That is the example we focus on in this case, since it covers just about all parts of what can be done this way.

The example:

```
POST /1/workflows/ HTTP/1.1
Host: rest.bosbec.io

Content-Type: application/json
api-key: 2992e4ab-5cf8-4343-94dc-50a4881cb0cb

{
    "TriggerNames": "t1",
    "WorkflowId": "4791b09c-fa35-4f81-90b9-c789e164b9e5",
    "MetaData": {
        "operation": "get-unit",
        "search": "+467012345"
    },
    "RequestSettings": {
        "content-type":"json",
        "ResponseData": {
            "output": "metadata.result",
            "output2": "incomingunit.metadata.test1"
        }
    }
}
```

To begin with an explanation here;

- In this API we can provide authentication-token in a header instead of in the JSON-data in the body.
- Must specify Content-Type : application/json
- The **TriggerNames** – parameter can be used to control what ExecutingWorkflow trigger to execute, provide a comma-separated list of trigger-names or just a single name in the string/text

- The **workflowId** – parameter will probably be able to be provided in the URL in the future, but for now it is the provided in the request body, and is the id of the workflow to execute (find in the details for your workflow in admin-gui)
- The **MetaData** - is the data you want to add/overwrite the current workflow-metadata with when the execution starts.
- The **ResponseSettings** – is something that is unique to this API, providing this property will cause the workflow to run directly during the request and return as soon as the execution is completed. **NOTE: Some jobs may not execute this way, for example ScheduleNextJobs.**
    - In the ResponseSettings you should specify **Content-Type** as "json" to get the response as JSON-data.
    - If you provide the **ResponseData** you may define what workflow context resources you want to get and what you want the name of that property should be in the responses data. (Example of response below)

```
{
    "data": {
        "output": "Some Unit,+46701234567,some.unit@bosbec.io",
        "output2": ""
    },
    "processId": "82e673de-360a-4289-823c-344ea88c7fa4",
    "executionTime": 267
}
```

Explained:

The workflow would in this case have a setup that could produce different outputs depending on what "**operation**" was provided in the metadata when posting the request to execute.
The example-result above was the result of finding a unit with the data from search, and putting some information from the unit in the **metadata.result** and then have it returned in the output property inside the responses **data**.
Another thing to say it that we've requested **output2** to be the data from the **incomingunit.metadata.test1**, but there were no such data available when the workflow execution was completed, and thus we have no data returned in that parameter. (A tip/hint here is to go in and have a look at the workflow context in admin-GUI to understand what is available when the execution was over)

The most accepting endpoint yet (in.bosbec.io)

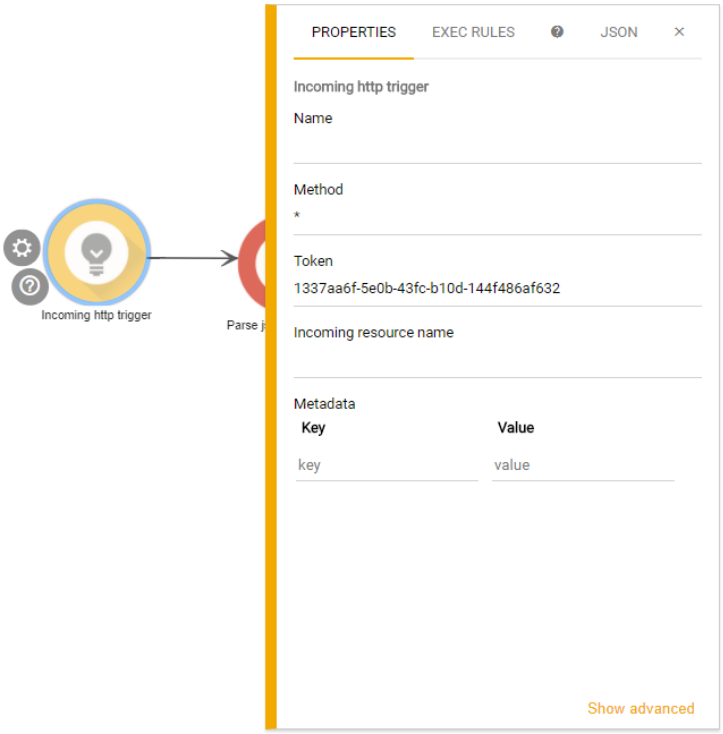
During the development of the different APIs mentioned above, there seem to be one part missing. We cannot be sure that every system that wants to integrate with Bosbec We always have the capabilities to format their requests after the needs in Bosbec We APIs. Therefore, we've developed an API that will allow you to send HTTP-requests without much more requirements than valid HTTP.

When a request arrives at the in.bosbec.io endpoint the HTTP-triggers (can be created in workflow-builder) will try to match it and if it does, execute the workflow.

Get your tokens ready

It is described how to get a token in the first parts of the tutorials, have a look there if you need to remember how it's done.

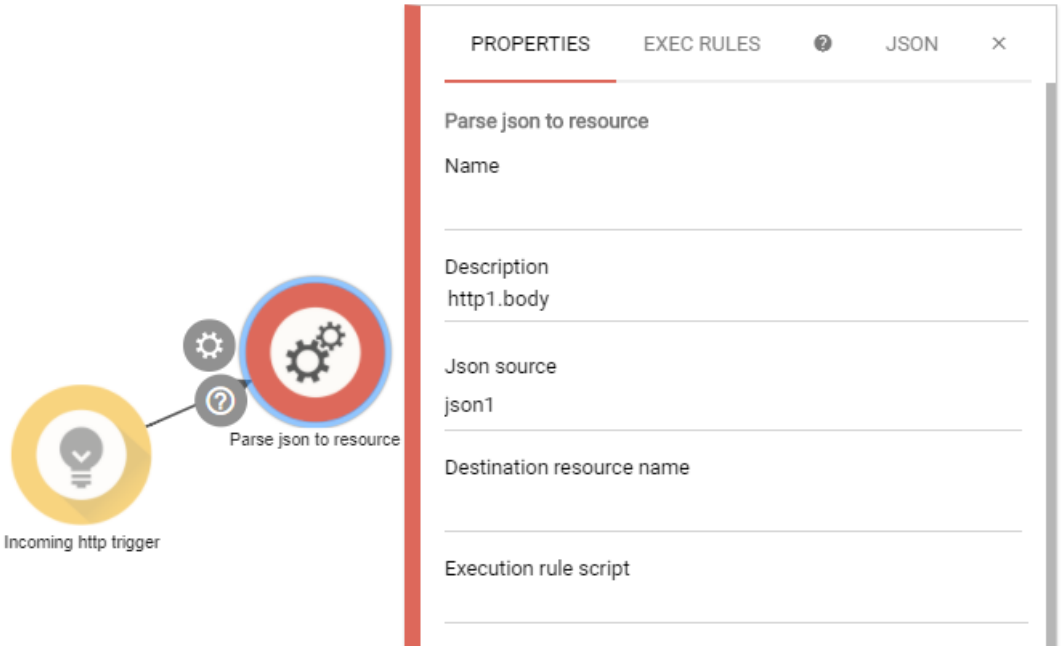
In this case you might want to create a new token, since this one is always sent in the URL, it acts both as your unique id with permissions and so on, but also as a uniquely reserved part of the URL. Assuming that your token is 1337aa6f-5e0b-43fc-b10d-144f486af632 then you could post requests to https://in.bosbec.io/1337aa6f-5e0b-43fc-b10d-144f486af632
And now at the time of writing this (July 2017) the trigger will only support GET or POST (or either one of those two methods) to start execution of the workflow.



Step1. Create an IncomingHttpTrigger and set the token and a name for your "http-resource" that you will use to refer to the http-request from jobs during the workflow execution.

In the example above, we've set the method to * so that we allow both GET and POST requests to trigger our workflow.

After this the next jobs will have access to **http1.body** (*just like a "normal" metadata-resource*) or **http1.header.content-type** (*just like a "normal" metadata-resource*).



Step2. If we are posting JSON-data into the in.bosbec.io

We have a job that could parse data from a resource (*like metadata.my-data or in this case http1.body*) and if we are to post JSON-data to our workflow we can make use of this to parse the JSON and then put that JSON-object in another resource (*called json1 in this case*).

How would we use **json1** in this case then? It's easy!
Assuming that we have a simple json like

```
{"some-data": 123}
```

Then we can use this new resource that we have parsed and from a job refer to it as **json1.some-data** and we would get the result **123**

Requests reference

In this section, we have gathered requests for the different URLs.

The requests below are copies of the requests made with a tool that we use to manually create requests and test out Bosbec We APIs. The tool is Postman (www.getpostman.com), of course it is possible to use whatever tool or skillfully crafted application of your own choosing to do this on your own 😊

The example requests start with information about what type of HTTP-request we are dealing with (however most of our API calls will use the POST method)

Then follow the headers (*the host header isn't required by the system but included in the examples since it is automatically added and provide useful information to what URL we are dealing with in the example)*.

Finally, we have the data sent. In most cases this is the JSON-data in POST HTTP-requests.

Authenticating via API2

https://api2.bosbec.io/authenticate

```
POST /authenticate HTTP/1.1

Host: api2.bosbec.io
Content-Type: application/json

{
    "data": {
        "username": "YOUR_USERNAME_HERE",
        "password": " YOUR_PASSWORD_HERE"
}
```

And the response would look like the example below.

Note that the part of the token-object to keep is the **Id** (**found in data.id**, *since the id in the root of the JSON-object is just the id of your request/response*)

```
{
    "data": {
        "id": "1337aa6f-5e0b-43fc-b10d-144f486af632",
        "issuedOn": "2017-07-01T06:21:34",
        "administratorId": "c8c40bf2-6897-48fa-a9e0-a0d600fe5edb",
        "appUserId": "00000000-0000-0000-0000-000000000000",
        "accountId": "12345678-1122-3344-5678-abcdef012348",
        "expiresOn": "2017-07-01T06:41:34",
        "isPersistent": false,
        "isValid": true
    },
    "id": "12345678-0123-5678-ab33-a41c345f4bb0",
    "status": "Success",
    "time": "2017-07-01T06:21:34",
    "requestUrl": "http://api2.bosbec.io:8887/authenticate",
    "errors": []
}
```

Executing workflow with parameters via API2

https://api2.bosbec.io/workflows/execute-with-parameters

```
POST /workflows/execute-with-parameters HTTP/1.1

Host: api2.bosbec.io
Content-Type: application/json

{
    "data": {
        "WorkflowId": "525f61e3-b8dd-4a15-b59e-320ac665a767",
        "TriggerNames": "trigger_1",
        "MetaData": {
            "my-data-1": "This is input data to the workflow"
        }
    },
    "AuthenticationToken": "1337aa6f-5e0b-43fc-b10d-144f486af632"
}
```

And a short explanation for the response below would be that the response has a **ProcessId** and that is the id to use if you want to call the /processes/details to get information on what happened during the execution of your workflow. The rest of the response is standard response data from api2.

```
{
    "data": {
        "processId": "926ed039-86b2-4e05-bd97-74ed6ebc3d03",
        "additionalData": null
    },
    "id": "3e6c1a68-8148-4e9b-a7f9-92e22e2b9dad",
    "status": "Success",
    "time": "2017-07-01T06:22:34",
    "requestUrl": "http://api2.bosbec.io:8887/workflows/execute-with-parameters",
```

```
    "errors": []
}
```

Rest.bosbec.io

https:// rest.bosbec.io/1/workflows/

```
POST /1/workflows/ HTTP/1.1
Host: rest.bosbec.io

Content-Type: application/json
api-key: 2992e4ab-5cf8-4343-94dc-50a4881cb0cb

{
    "TriggerNames": "t1",
    "WorkflowId": "4791b09c-fa35-4f81-90b9-c789e164b9e5",
    "MetaData": {
        "operation": "get-unit",
        "search": "+46701234567"
    },
    "RequestSettings": {
        "content-type":"json",
        "ResponseData": {
            "output": "metadata.result",
            "output2": "incomingunit.metadata.test1"
        }
    }
}
```

The response below, for the given request above, is better explained in the tutorial-text in the first section of this document, but the short version is that a workflow executes and put some result in **metadata.result** and/or **incomingunit.metadata.test1** depending on the “operation” provided into the execution of the workflow. And this results in that the API returns data the way it has in this example.

```
{
    "data": {
        "output": "Some Unit,+46701234567,some.unit@bosbec.io",
        "output2": ""
    },
    "processId": "82e673de-360a-4289-823c-344ea88c7fa4",
    "executionTime": 267
}
```