Distributed Network Traffic Generator - 5G Networks project - server side documentation

[REST API] [WebSocket]

REST API

Header in every request (except */auth & /auth/login*) most contain:

```
headers: {
   "Authorization": `Bearer ${token}`,
},
```

Endpoint	Description	Request body
/auth	Used for creating account. Password should be 8 characters and contain at least: 1 uppercase letter, 1 special sign, 1 number. HTTP METHOD: [POST] 400 - Bad request - user already exists or password is too weak. 201 - Created	**Request body:** `{ "login": "string", "password": "S@molocik12345" }` **Response body:** `{ "id": "long" "login": "string", "role": "string", "token": "string" }`

/auth/login	Used for logging in to account. HTTP METHOD: [POST] 400 - Bad request - login not found or validation error 200 - Created	**Request body:** `{ "login": "string", "password": "string", }` **Response body:** `{ "id": "long" "login": "string", "role": "string", "token": "string" }`
/auth/device/login	Used for logging in with device. Sending login, password and name **creates new device.** Sending login, password and uuid **logs device to the system.** HTTP METHOD: [POST] 400 - Bad request - username not found, validation error or not linked device 200 - Ok	**Request body:** `{ "login": "string", "password": "string", "uuid": "string", "name": "string" }` **Response body:** `{ "token": "string", "uuid": "string" }`

/device	Returns list of devices fullfiling specified search criteria. You can search by: • uuid, • device type: ◦ “IOS”, “LINUX”, “ANDROID” ◦ you can search by more than one device type at once, • device status: ◦ “ACTIVE”, “WORKING”, “INACTIVE”, ◦ you can search by more than one device status at once, • device name, or sort by: • uuid descending - “UUID_DESC”, • uuid ascending - “UUID_ASC“, • name descending - “NAME_DESC”, • name ascending - “NAME_ASC“ Example request: `/device?deviceTypes=IOS&deviceTypes=LINUX&deviceStatuses=WORKING` HTTP METHOD: [GET] 200 - Ok	**Params:** “uuid”, “deviceTypes”, “deviceStatuses”, “name”, “sortBy” **Response body:** ``` [{ "uuid": "string", "deviceType": "string", "deviceModel": "string", "lastConnection": "string", "status": "string" }, ...] ```

/device/active	Informs server that device is active. **This request must be sent once every 10 minutes. Otherwise, the device will be considered inactive.** HTTP METHOD: [POST] 400 - Bad request 200 - Ok	**Request header:** `headers: { "Content-Type": "json /application", "Authorization": `Bearer ${token}`, "device-uuid": uuid, },` **Response body:** `{ "uuid": "string", "deviceType": "string", "deviceModel": "string", "lastConnection": "string", "status": "string" },`
/device/disconnect	Informs server that device is inactive HTTP METHOD: [POST] 400 - Bad request 200 - Ok	**Request header:** `headers: { "Content-Type": "json /application", "Authorization": `Bearer ${token}`, "device-uuid": uuid, },` **Response body:** `{ "uuid": "string", "deviceType": "string", "deviceModel": "string", "lastConnection": "string", "status": "string" },`

/device/tasks	Returns list of tasks for device specified by uuid. HTTP METHOD: [GET] 400 - Bad request - invalid device type 200 - Ok	**Request header:** `headers: { "Content-Type": "json /application", "Authorization": `Bearer ${token}`, "device-uuid": uuid, },` **Response body:** `[ { "id": "long", "taskType": "string", "status": "string", "fileUrl": "string", "orderStart": "string", "orderEnd": "string", "device": { "uuid": "string", " deviceType": "string", "deviceModel": "string", } }, ... ]`

<table>
<tr>
<td>device/tasks/{taskId}
/upload</td>
<td>Uploads file.

HTTP METHOD: [POST]

400 - Bad request

200 - Ok</td>
<td>Request header:

<pre>
headers: {
   "Content-Type": "multipart
/form-data",
   "Authorization": `Bearer
${token}`,
   "device-uuid": uuid,
},
</pre>
Path variable:

“taskId” must be passed as path variable.

Request body:

<pre>
{
  "file": file,
}
</pre></td>
</tr>
<tr>
<td>/tasks</td>
<td>Used for creating tasks.

Task types:

• “STREAMING”,
• “SFTP”,
• “HTTP”

HTTP METHOD: [POST]

400 - Bad request

200 - Ok</td>
<td>Request body:

<pre>
{
  "uuid": "string",
  "type": "string",
}
</pre>
Response body:

<pre>
{
  "id": "long",
  "taskType": "string",
  "status": "string",
  "fileUrl": "string",
  "orderStart": "string",
  "orderEnd": "string",
  "deviceGetDto": {
    "uuid": "string",
    "deviceType": "string",
    "deviceModel": "string",
  }
}
</pre></td>
</tr>
</table>

<table>
<tr>
<td>/tasks</td>
<td>Returns list of devices fullfiling speified search criteria.

You can search by:

• Task types:
  ◦ “STREAMING”, “SFTP”, “HTTP”,
  ◦ you can search by more than one task type at once,
• device uuid,,
• device type:
  ◦ “IOS”, “LINUX”, “ANDROID”,
  ◦ you can search by more than one device status at once,
• task status:
  ◦ “NEW”, “IN_PROGRESS“, “FINISHED”,
  ◦ you can search by more than one task status at once,
• device name,

Example request:

<code>/tasks?deviceTypes=IOS&deviceTypes=LINUX&taskStatuses=FINISHED</code>

HTTP METHOD: [GET]

400 - Bad request

200 - Ok</td>
<td>Params:

“taskTypes”, “deviceUuid”, “deviceName”, “taskStatuses”, “deviceTypes”

Response body:

<pre>[
  {
    "id": "long",
    "taskType": "string",
    "status": "string",
    "fileUrl": "string",
    "orderStart": "string",
    "orderEnd": "string",
    "device": {
      "uuid": "string",
      "deviceType": "string",
      "deviceModel": "string",
    }
  },
  ...
]</pre></td>
</tr>
<tr>
<td>/tasks/{taskId}
/download</td>
<td>Downloads file.

HTTP METHOD: [GET]

400 - Bad request

200 - Ok</td>
<td>Path variable:

“taskId” must be passed as path variable.

Response body:

You receive file.</td>
</tr>
</table>

WebSocket

Connection

User WS connection address:

`ws://localhost:8010/userws`

User WS connection task address:

`ws://localhost:8010/devicews`

Device WS connection address:

`ws://localhost:8010/devicews`

To connect you also need pass authorization token in the header:

```
headers: {
   "Authorization": `Bearer ${token}`,
},
```

Subscription:

You can subscribe to the following topics (For devices):

- `/topic/device_${uuid}`
  - `uuid` - device uuid
  - from this topic you receive information about task you need to execute:

```
{
   "taskId": "long",
   "taskType": "string",
 }
```

- `/topic/user_${user_id}`
  - `user_id` - userId
  - from this topic you receive:

```
{
   "taskId": "long",
   "taskType": "string",
   "status": FINISHED
 }
```

- `/topic/global`
  - from this topic you receive device status message:

```
{
   "uuid": "string",
   "deviceModel": "string",
   "deviceType": "string",
   "status": "string"
 }
```

Example of connection and subscription

with JavaScript library @stomp/stompjs

```
var client = Stomp.client("ws://localhost:8010/userws");
client.connect(
  {
    Authorization:
      `Bearer ${token}`,
  },
  function () {
    client.subscribe(
      `/topic/user_${id}`,
      function (message) {
        console.log(message.body);
      },
    );
  }
);
```