This description is for reference only. All type of websocket packets and Rest API, formats, headers, encryption and other will be discussed and added new features if it's necessary.

Rest API description:

- About Encryption

All HTTP data in TCP/IP packets will be encrypted with a static AES-128 key: "AgrocityAgrocity" =

0x41, 0x67, 0x72, 0x6f, 0x63, 0x69, 0x74, 0x79, 0x41, 0x67, 0x72, 0x6f, 0x63, 0x69, 0x74, 0x79 (excepts API`s for administrator role)

This is an only a test key, release key will be determined. So, it needs to put somewhere in configuration file of server for next changes.

- About Headers

All API will contain next minimum headers:

User-Agent: device/version Example: User-Agent: AdvancedGpsTrackerRev1.0/256

Comment:

For future, we can have many types of devices and this header is for its identification

Authorization: we can use **Basic Auth** or **Bearer Token**

Authorization: Basic {{base64encrypt(user:pass)}} Example: Authorization: Basic dXNlcjpwYXNz

Authorization: Bearer {TOKEN} Example: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcGlfa2V5X2lkIjoiNjFlMTU4YWUtYTNhYy00NzNmLWIwNGUtMzdlNWNkZDVkM2E5IiwiYXVkIjoiYXMiLCJpc3MiOiJhcyIsIm5iZiI6MTYzMDU4OTY1MCwic3ViIjoiYXBpX2tleSJ9.t9eEaPhsQqVsBRlQAhOSQrjdAy2rLsJEF1Lo7YmTYNE

Comment:

Authorization type will be determined. At my opinion Basic is enough because all HTTP data will be encrypted with a static AES-128 key and we need to take into consideration traffic consumption. Each device will have a personalized user/pass or Bearer Token stored in DB for its identification by server

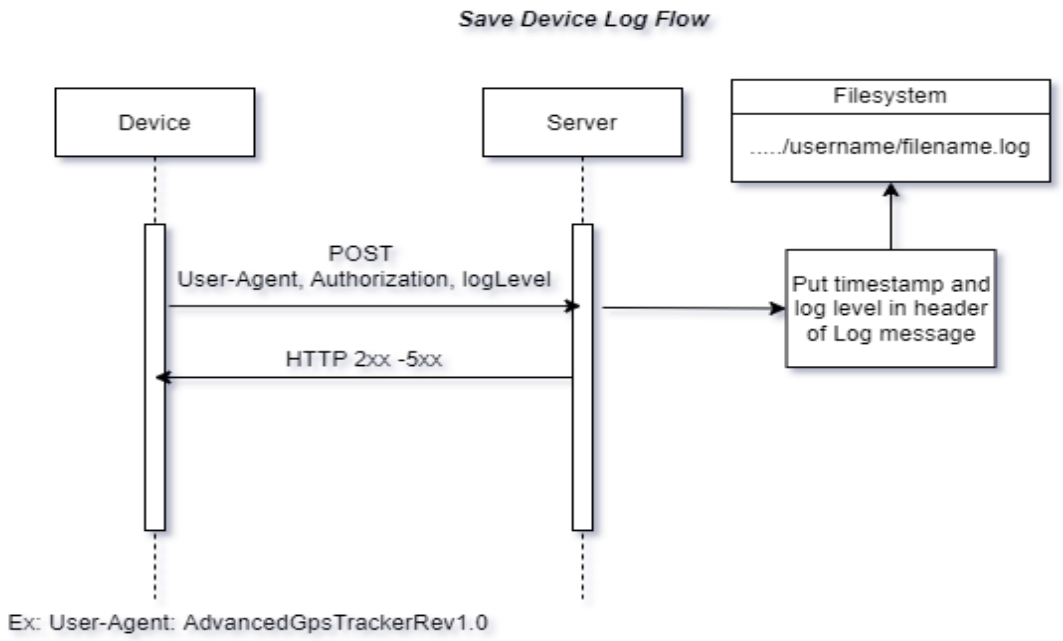
Save Device Log:

Request type: ***POST***

{{baseUrl}}/api/device/log/:logLevel

logLevel = {ERROR, WARN, DEBUG, INFO, TRACE}

body will contain a raw message



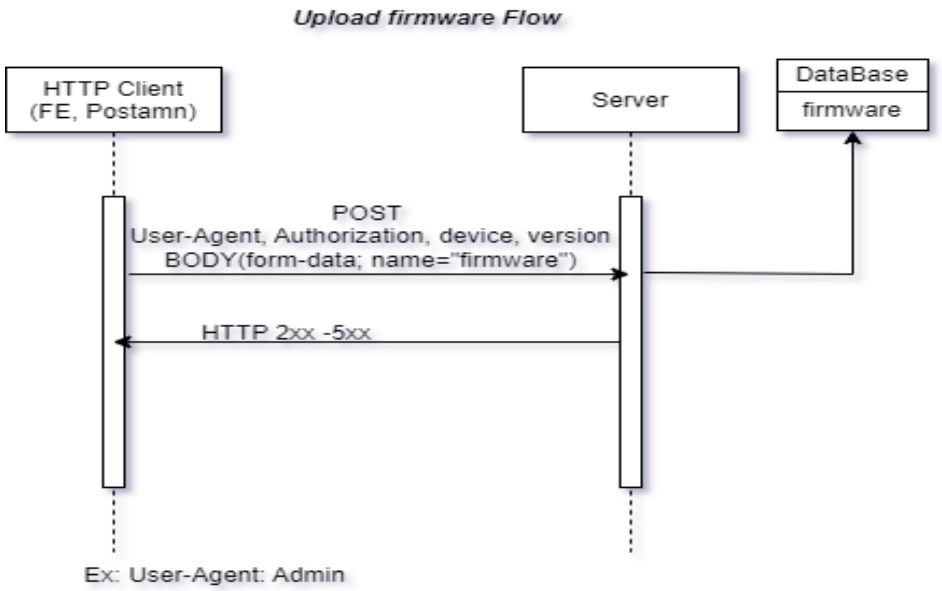
Comment:

This is an additional method to send log to server.
This method was introduced, because if any problem occur with websocket connection device will be able to send a log message to a rest API, and information about errors will be available in log. By default, logging method will be used through websocket because it uses less traffic. Before to add a message to a log file, server need to add at the front of message date and time (the device can send message with date/time but we need to reduce traffic at maximum). Example [dd/mm/yyyy] [hh:mm:ss][logLevel] -> received raw message.

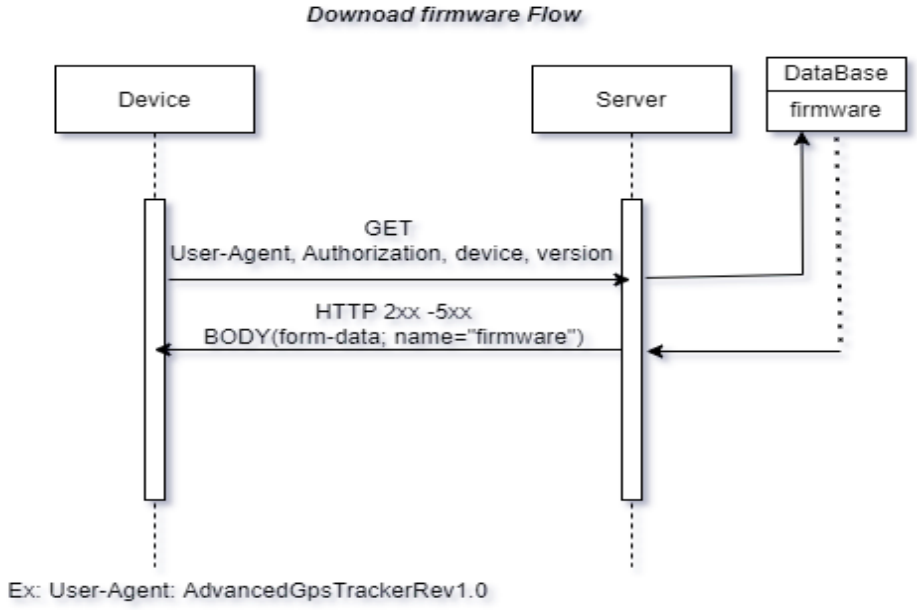
Firmware:

Request type:

POST – upload firmware (use HTTPS not static encryption, will be used by FE or ex. Postman)



GET – download firmware (static encryption, used by devices)



{{baseUrl}}/api/device/firmware/:device/:version

device = AdvancedGpsTrackerRev1.0, we can have many types of devices and server need to be transparent for this parameter

version = unsigned integer 2 bytes in decimal notation

MSB – major version

LSB – minor version

Example: Version 1.0 = 0x01, 0x00 = MSB<<8|LSB = 256 in decimal notation

{{baseUrl}}/api/device/firmware/AdvancedGpsTrackerRev1.0/256

Comment:

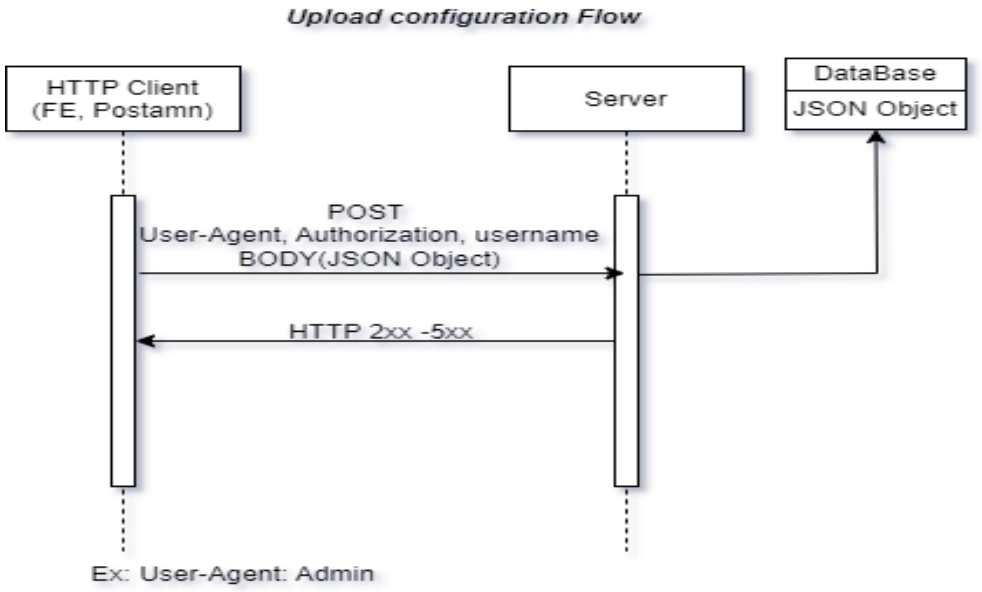
- Upload is allowed only for administrator role, will be determined
- Download is allowed for any devices and users

Configuration:

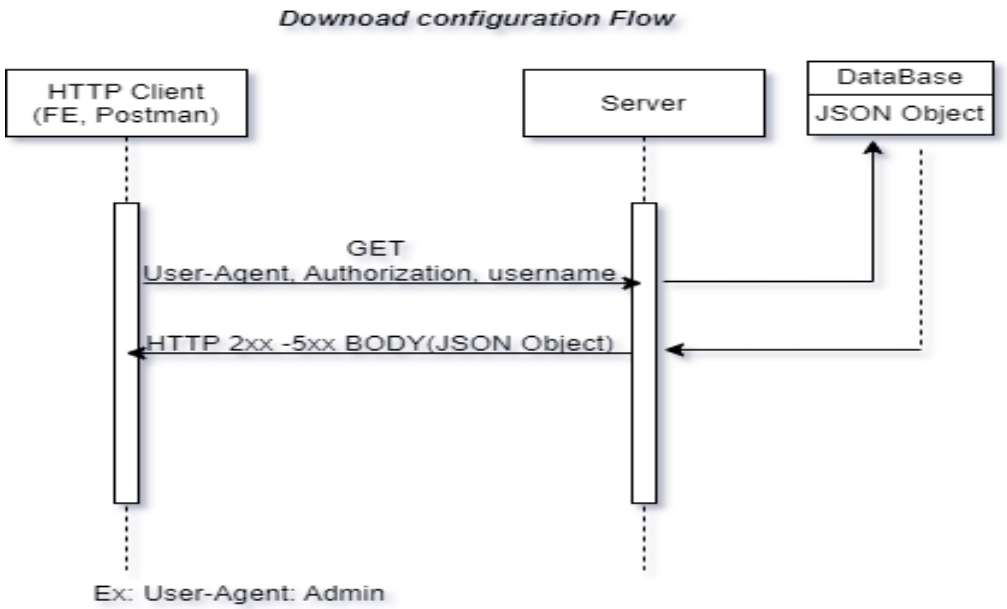
This is a method to upload/download configuration to/from DB for FE or Postman

Request type:

POST – upload configuration (use HTTPS not static encryption, will be used by FE or ex. Postman)



GET – download configuration (use HTTPS not static encryption, will be used by FE or ex. Postman)



{{baseUrl}}/api/device/config/:username

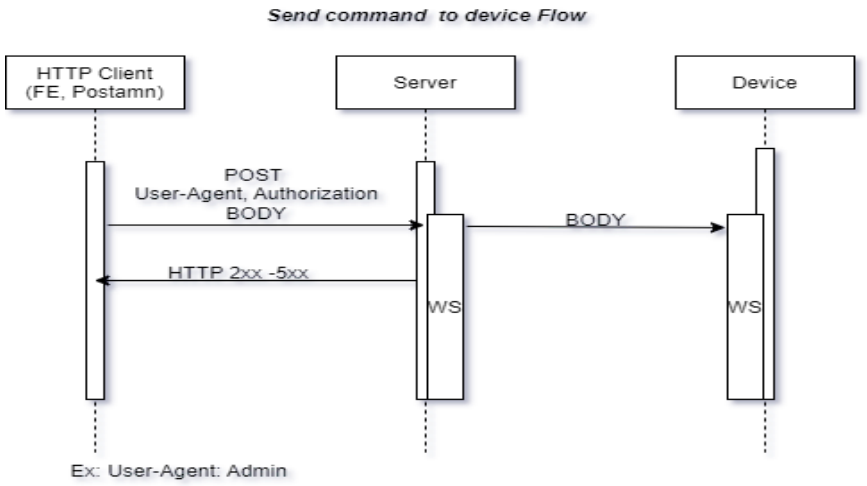
username = a username config destination

Comment:

- Upload is allowed only for administrator role, will be determined
- Download is allowed for any devices and users

Send Command to device:

Request type: ***POST*** (use HTTPS not static encryption, will be used by FE or ex. Postman)



{{baseUrl}}/api/device/command/:username

username – user name destination for this message (if username = bulk then server need send this message to all devices) If registered device doses not have an active session, return HTTP 4xx-5xx else HTTP 200. Body will contain a JSON message which will be forwarded to device through websocket. This API is available only for administrator role. Will be used for device configuration and updates.

Websocket:

Request type: ***GET***

Standard websocket protocol

Through websocket connection will be transmitted commands and data from server to device and backward in JSON format

JSON protocol description:

{Packet_type:{Packet_Info}}

Packet_type:

1. "***Command***" command to server/device, will contain a string command or object in case of "Version"
2. "***Config***" configuration data will contain an object with configuration
3. "***Data***" data from device to server will contain an object with some data or integer
4. "***Log***" log messages from device to server
5. "***Info***" all disponible static information about device ex. (Type of modem, IMEI, UID… will be determined)
6. "***Status***" all disponible status information about device

1. Command packet:

Commands from server to device:

1.1. {"Command":"***Reboot***"} – device will be rebooted

1.2 {"Command":"***ConfigPut***"} – device will send a Config packet to server with actual config

1.3 {"Command":"***ConfigGet***"} – device will get from server stored configuration

1.4 {"Command":"***ConfigApply***"} – device will save new received config to flash.

1.5 {"Command":{"***Version***":unsigned int}} - the last available version of firmware on server for this type of device based on ***User-Agent: device/version*** header when websocket connection was initiated.

1.6 {"Command":"***DataGet***"} – device will send all available data to server

1.7 {"Command":{"***DataRateSet***":unsigned int}} – data rate 0-65535 seconds, if 0 – data will be delivered to server on request(see 1.5 ***DataGet***), 1-65535 – delay between packets.

1.8 {"Command":"***InfoGet***"} – device will response with **Info Packet**.

1.9{"Command":"***StatusGet***"} - device will response with **Info Packet**.

1.10{"Command":{"***VLSet***":unsigned int}} – set verbose level 0-255. If 0-loging is disabled. Default verbose level device get from config, this command is only for debug.

Bit	7	6	5	4	3	2	1	0

	RES3	RES2	RES1	INFO	TRACE	DEBUG	WARN	ERROR
Value	X	x	x	0/1	0/1	0/1	0/1	0/1

X – does not matter 0 or 1

0 – logging for this event disabled

1 – logging for this event enabled

Example: for enable logging only for errors {"Command":{"***SetVL***":1}}

Commands from device to server:

1.11 {"Command":"***ConfigGet***"} – server need to respond with "***Config***" packet where in object are contained configuration from DB stored for this device(not device type but by UID or username).

2. Config Packet:

{"Config":***Object***} – this packet is allowed from device to server and from server to device. It contains a configuration in ***Object.***

Example:


```
{
 "Config": {
  "Device_ID": "UID_002F003E3038510936333335",
  "Device_MAC": "DE:AD:BE:EF:00:01",
  "Device_Type": "AdvancedGPSTracker",
  "Device_HW_Version": [0, 1],
  "IP_Address": [192, 168, 11, 1],
  "Mask": [255, 255, 255, 0],
  "DHCP_Client": true,
  "DHCP_Server": false,
  "APN": "orange.ro",
  "PIN": "1873",
  "PPP_MAX_SPEED": 921600,
  "AT_Timeout": 10,
  "MQTT_Server_port": 1883,
  "Credentials": [
   {
    "Username": "User0",
    "Password": "Password0"
```

```
    },
    {
      "Username": "User1",
      "Password": "Password1"
    },
    {
      "Username": "User2",
      "Password": "Password2"
    },
    {
      "Username": "User3",
      "Password": "Password3"
    }
  ],
  "APP_Server_address": "https://iello.tech",
  "APP_Server_port": 443,
  "APP_SSL_enabled": false,
  "Certs_Path": "\\certificates",
  "Manual_DNS": false,
  "DNS1_IP": [8, 8, 8, 8],
  "DNS2_IP": [8, 8, 4, 4],
  "NTP_Server": "time.google.com"
 }
}
```


3. *Data Packet:*

{“Data”:***Object***} or {“Data”: unsigned integer}

Object contains some data (timestamp, location, speed, status, weigth… will be determined format and content)

In case of 4G connection lost, device will store all data in NVRAM and after reconnection to server all data will be transmitted from NVRAM to server, in this case in Object will be included a one additional key to make a distinction between historical and live data (will be determined this thing)

If server receive a {“Data”:unsigned integer} this mean no any changes in Data and last packet is valid, integer will contains only actual timestamp. This type of packet needed for minimize data traffic through network when not have “significative changes” in data.

“Significative changes” - will be determined

4. *Log Packet:*

{“Log”:***Object***} - this packet is allowed from device to server. It contains in ***Object***

Object key = {“ERROR”, “WARN”, “DEBUG”, “INFO”, “TRACE”}

Object value = string, the message itself.

Example:


```
{
 "Log": {
  "ERROR": "Can't initiate websocket connection\r\n"
 }
}
```


5. *Info Packet:*

{“Info”:***Object***} - this packet is allowed from device to server. It contains an ***Object with*** all disponible static information about this device (will be determined format and content)

Example:

Response to “***InfoGet***” command:


```
{
 "Info": {
  "Device_ID":  "UID_002F003E3038510936333335",
  "Device_MAC":  "DE:AD:BE:EF:00:01",
  "Device_Type":  "AdvancedGPSTracker",
  "Device_HW_Version":  [0, 1],
  "Device_FW_Version":  [0, 1],
  "IP_Address":  [192, 168, 11, 1],
  "4G_Modem_Model": "SIMCOM_SIM7600C",
  "4G_Modem_Rev": "LE11B01SIM7600C",
  "4G_Serial_Number": "351602000330570",
  "SIM_IMSI": "460010222028133",
  "SIM_ICID": "898600700907A6019125",
  "SIM_IMEI": "357396012183170",
  "SIM_SPN": "Orange"
```

```
 }
}
```

6. *Status Packet:*

{“Status”:***Object***} - this packet is allowed from device to server. It contains an ***Object with*** all disponible status information about this device (will be determined format and content)

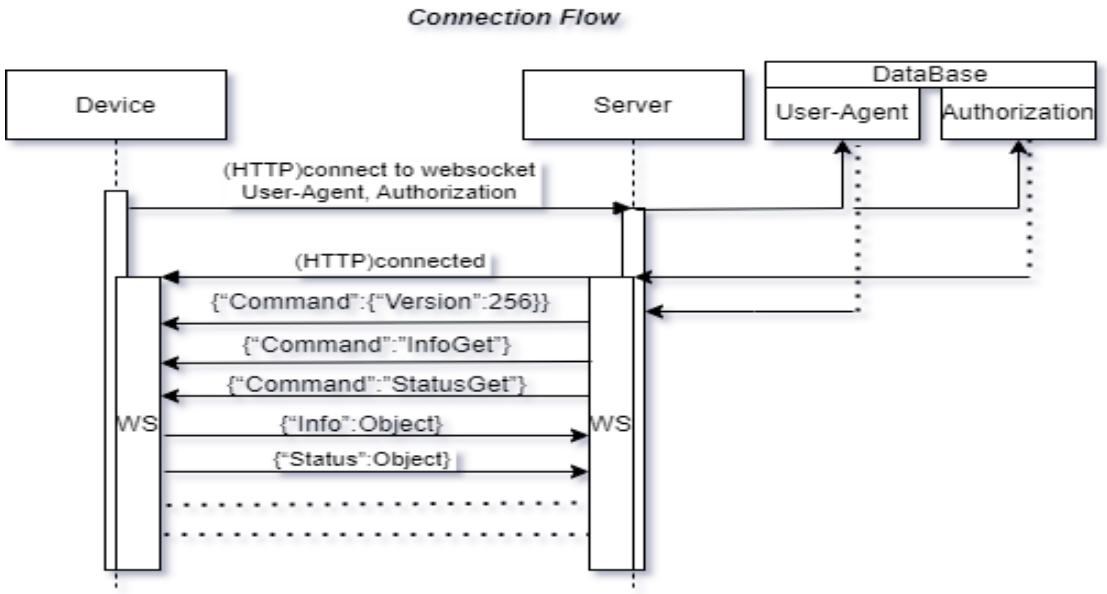
Example:

Response to “***StatusGet***” command:

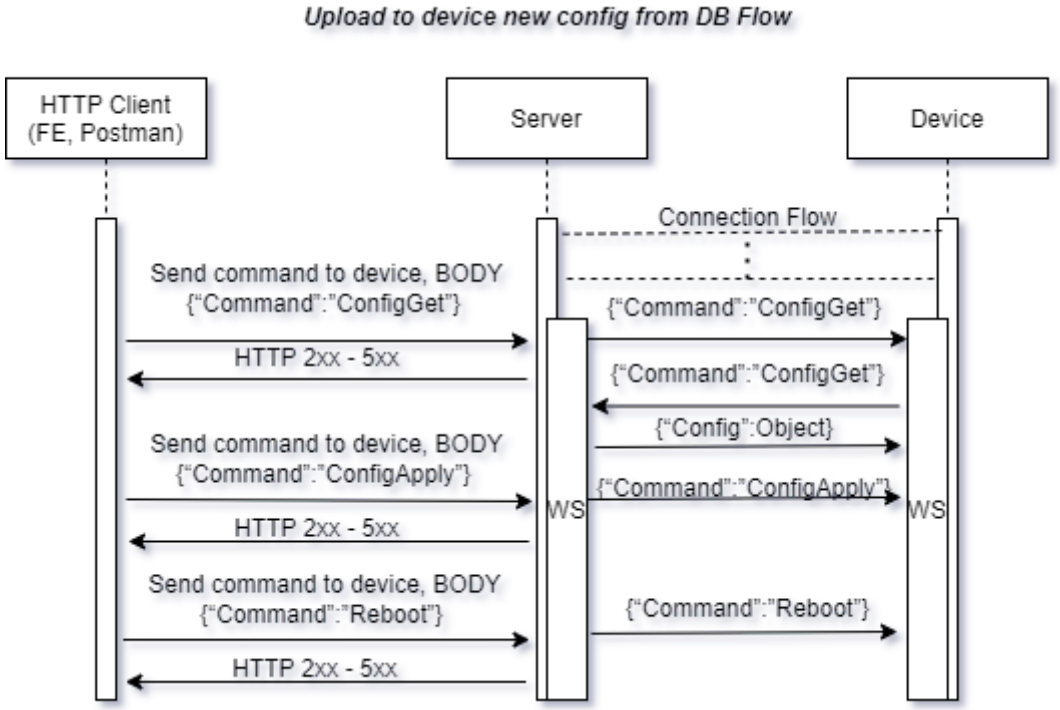
```
{
 "Status": {
  "4G_Modem_Mode": "LTE",
  "4G_Modem_CSQ": 72,
  "4G_IP_Address": [10, 0, 0, 10]
 }
}
```

Connection Flow

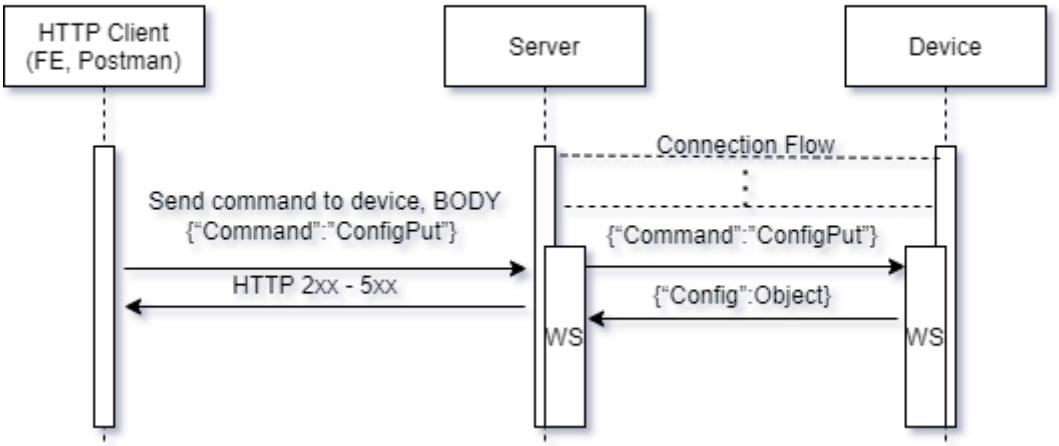
1. System at start will try to connect to server through websocket (IP or domain name and connection port gets from configuration). In webscocket connection headers server will be receive “***User-agent:***” and “***Authorization:”*** headers.
2. When websocket is initialized, server need to send a command packet with latest available version on the server for device contained in “***User-agent:***” value, ***InfoGet*** and ***StatusGet*** command are optional.
3. If command packet with latest available version will contain a newer version then device will perform firmware update and will be restarted, else working flow will be continued.



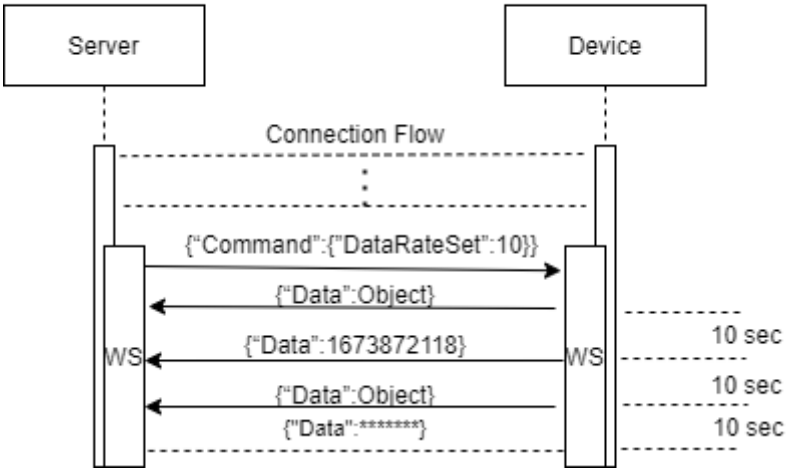
Working Flow



Download from device actual config Flow



Set data rate to 10 seconds Flow



Set data rate to 0 Flow