Nuki Web API

Nuki APIs - Terms of Use
Version v.1.5.3
12.11.2024

INTRODUCTION

Welcome to the Nuki Web API documentation. This API lets developers control and manage Nuki devices like Smart Locks, Smart Doors, and Openers remotely. These devices need to be connected to the internet either through a Nuki Bridge or built-in Wi-Fi.

The Nuki Web API securely sends commands to devices via HTTP/TLS connection, either through the Nuki Bridge or directly to the Smart Lock Pro's Wi-Fi. Responses are then sent back to Nuki Web.

All commands are executed using the server-stored Nuki Web Authentication Key. This means Nuki Web works independently of other clients, like the Nuki App for iOS and Android.

At a very high level, to access the Nuki Web API:

- Connect Smart Lock to a Nuki Bridge, or use the in-built Wi-Fi in case of Pro devices to bring the devices online
- Activate Nuki Web for the devices from the Nuki App
- Obtain an Authentication Key through one of the authentication methods (API token or OAuth 2)
- Use this Authentication Key to integrate with the Nuki Web API

For detailed instructions, please refer to the Quick Start Guide in the following section.

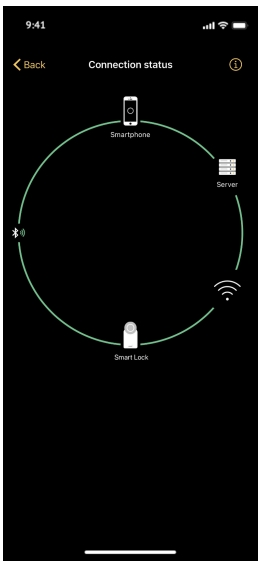
QUICK START GUIDE

Learn how to easily connect and control your Nuki devices with the Nuki Web API.

1. Setup the device

The Nuki Web API can only be tested with real Nuki devices connected to a Nuki Web account. Hence, first setup your Nuki devices in the Nuki App.

The devices should have online access. Thus, connect your devices to your WiFi via a Nuki Bridge or use the built-in WiFi of the Smart Lock Pro. Once the device is connected, you can validate this in the Nuki App settings under “Connection status” where the device should be shown as connected to the server.

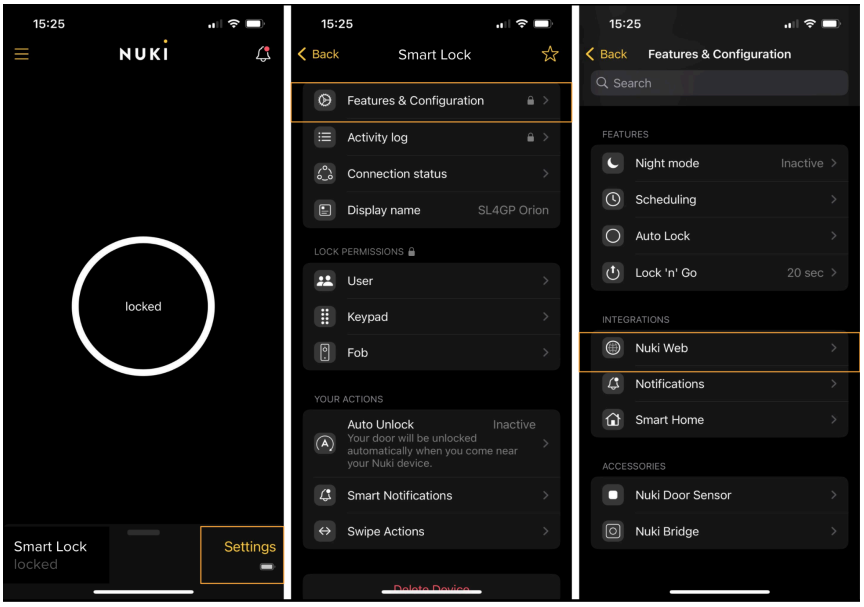


2. Connect the device to Nuki Web

Devices installed in the Nuki App can be connected to Nuki Web through an email address. The Nuki Web platform provides remote access to the devices and also the ability to be accessed via the Nuki Web API.

The steps to connect a device to the Nuki Web platform are:

- Tap on the desired Nuki device in the Nuki App and enter the “Settings”.
- Then go to "Features & Configuration". Here you can find the option "Activate Nuki Web". Follow the instructions to connect the device to your Nuki Web account.
- If you do not have an account yet, a new one can be created in the process. Refer to this article in the Nuki Help Center for more details.
- You can login to Nuki Web from here: https://web.nuki.io/

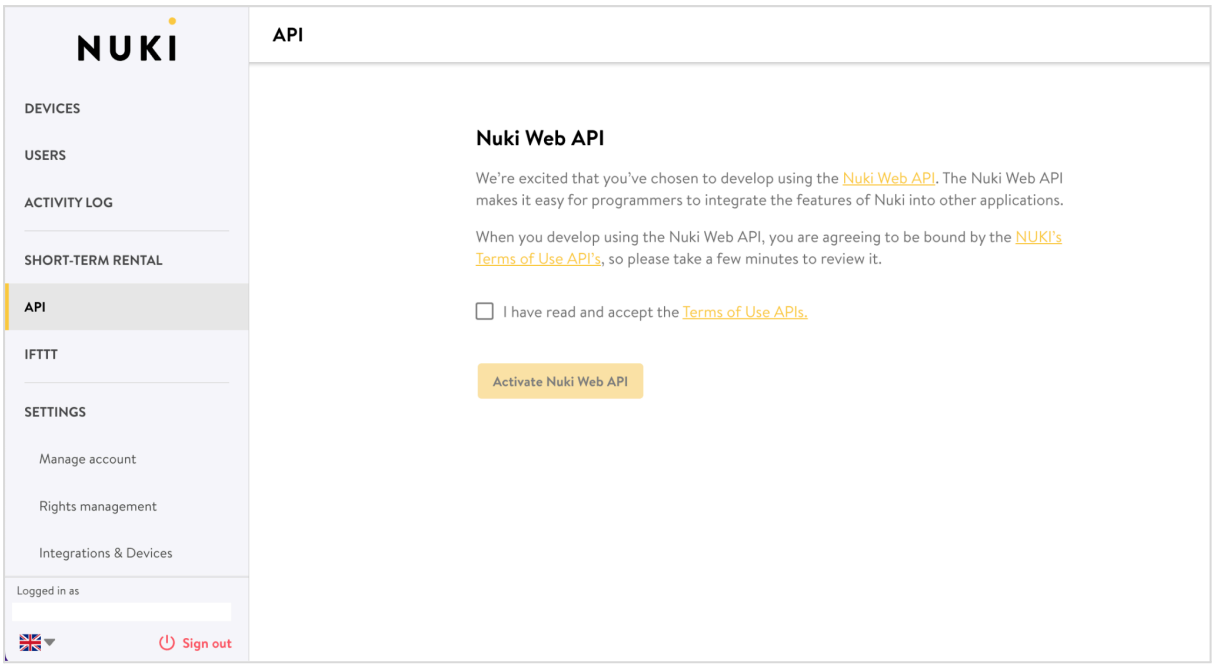


3. Activate the Nuki Web API

The Nuki Web API can be accessed from the Nuki Web platform.

The steps to connect to the Nuki Web API are:

- Login to Nuki Web with your registered email address.
- From the left-side Menu, access the “API” feature and accept the Terms of Usage of APIs to activate the API.

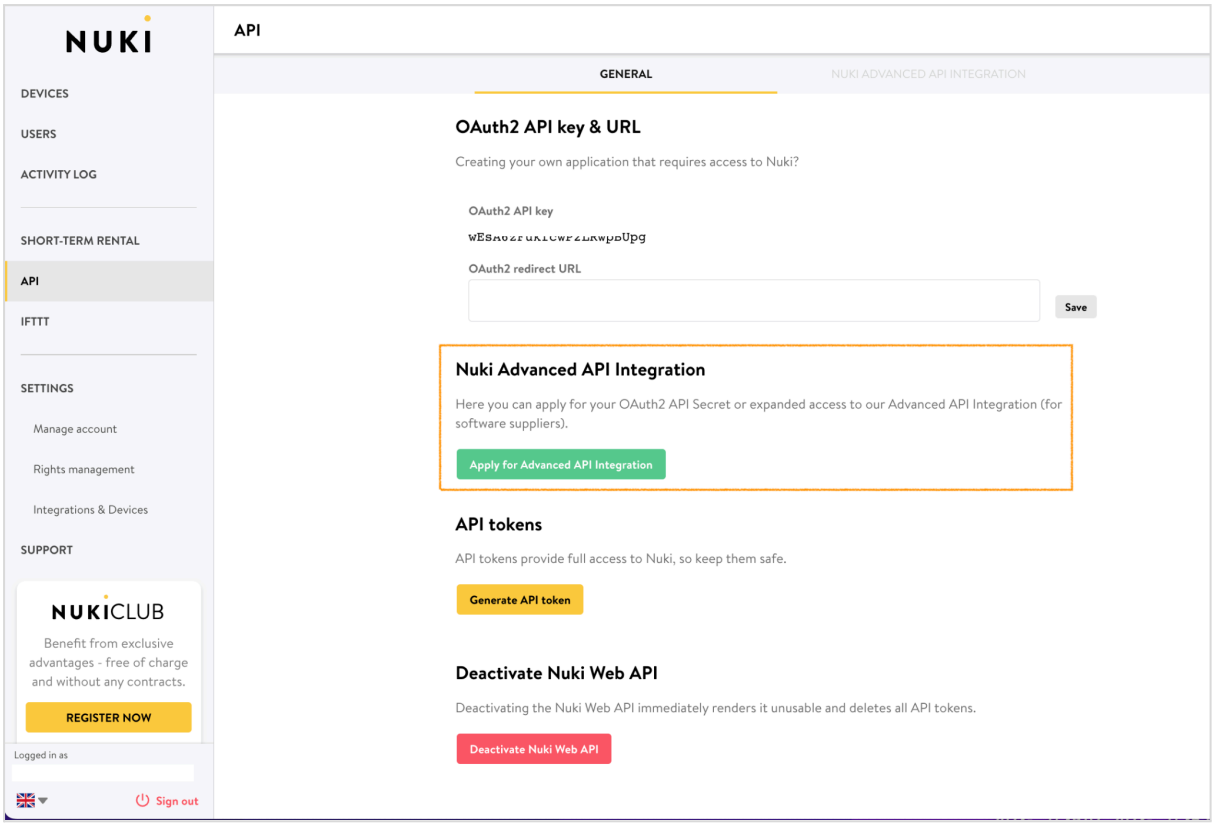


- This gives access to the “General” Nuki Web API.
- In order to use **webhooks** or **OAuth 2** for integration, you must apply for a request to use the Nuki Advanced API Integration. However, this is subject to your use case.

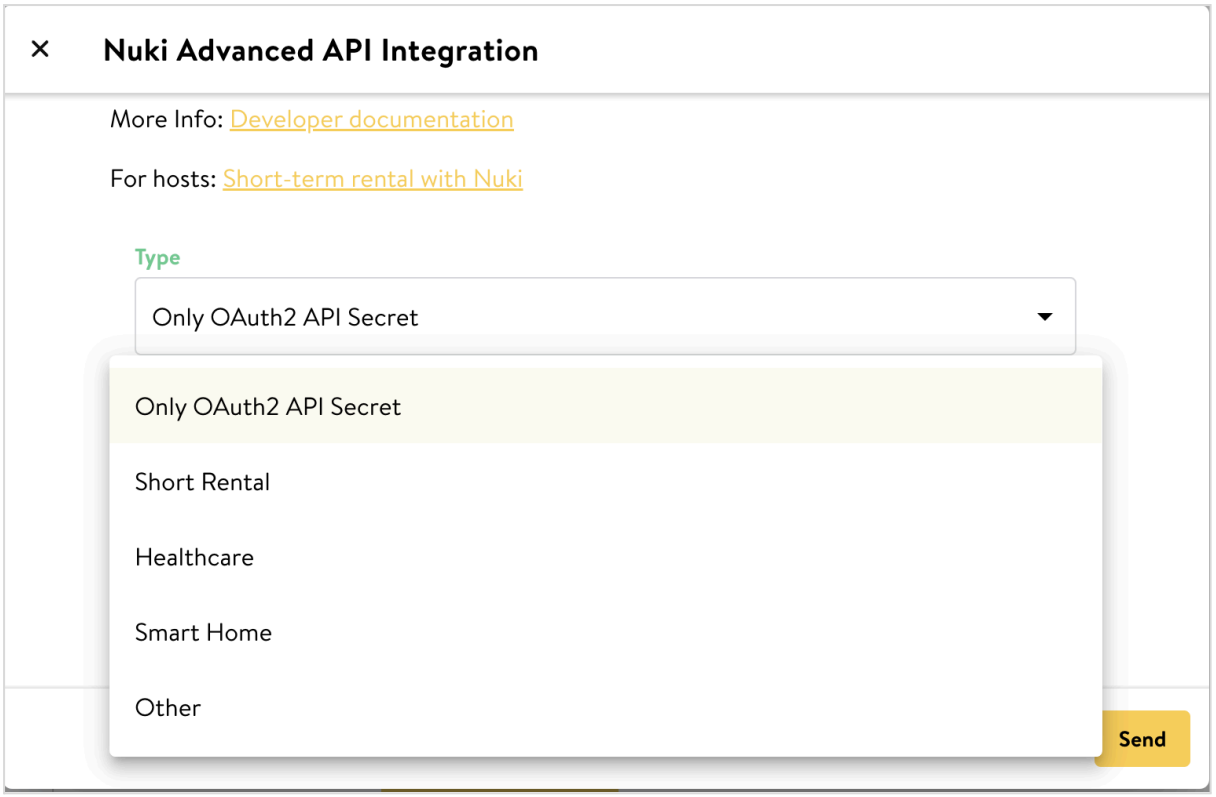
4. Apply for Nuki Advanced API access

In order to use **webhooks** or **OAuth 2** for integration, you must apply for a request to use the Nuki Advanced API Integration. The Nuki Advanced API can be accessed only if your request is approved by the Nuki Developer team.

- Login to Nuki Web, and access the API section. Here, under “Nuki Advanced API Integration”, you will have an option to send a request for Advanced API access.



- While creating the request, select the “Type” based on your use case. For example,
    - If you need to use the API for short rental purposes, then select “Short Rental”.
    - If you need the client secret for OAuth 2 integration but without webhooks, then select “Only OAuth2 API Secret”.
    - If you need to use the API for webhooks, and it is neither of the stated purposes, select “Other” type.



- While you provide the details such as name, email, please ensure that you provide an email that can be used to contact you and the Webhook URL you provide here must exactly match the one that you use.
- Once the request is approved by the Nuki Developer team, you should be able to see the “Nuki Advanced API Integration” section and the OAuth 2 secret.

5. Authenticate calls to the Nuki Web API

The calls to the Nuki Web API require appropriate authorization. This is done through the “bearer token” which should be present in every call made to the API. There are two ways to obtain a valid bearer token, which are described below. Integrators are suggested to adopt the OAuth 2 authentication type.

Types of authentication

#	Authentication Type	Description
1	OAuth 2	OAuth 2 (also referred to as OAuth 2.0) is the industry-standard protocol for authorization. Use this method • when you are offering an application to your end users which grants your application/server the right to operate their Smart Locks • when your users have no technical experience

		and you want to offer a simple login to your services without the need for the end user to generate API tokens and copy them around For example, a company providing short rental services for hosts should implement OAuth 2 to asks customers to login through Nuki Web account For more details on the OAuth 2, refer to the Core Concepts Section B: OAuth 2.0.
2	API Token	Use this method only when you use the Nuki Web API to access your own Nuki Web account, with your own Smart Locks For example, for personal use of the API integration with Home Assistant NOTE: We do not recommend going live with this method as we plan to sunset this feature in the near future.

1. OAuth 2 (Authorization Code Flow)

OAuth 2.0 is the industry-standard protocol for authorization. Authorization Code Flow is one of the grant types of OAuth 2 and it is the most secure and widely used OAuth 2 flow for web applications. Nuki Web uses OAuth 2 to grant client applications (by external integrators) access to Nuki Web users' devices without sharing passwords.

Below are the key steps for a client application to obtain the access token of a user through the OAuth 2 Authorization Code Flow:

1.1 Client application implements an authorization code link

The client application provides the user with a link or button as a starting point. Upon clicking the link or button, the client application has to send a request to the Authorization Server with the following parameters:

`https://api.nuki.io/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=SCOPES`

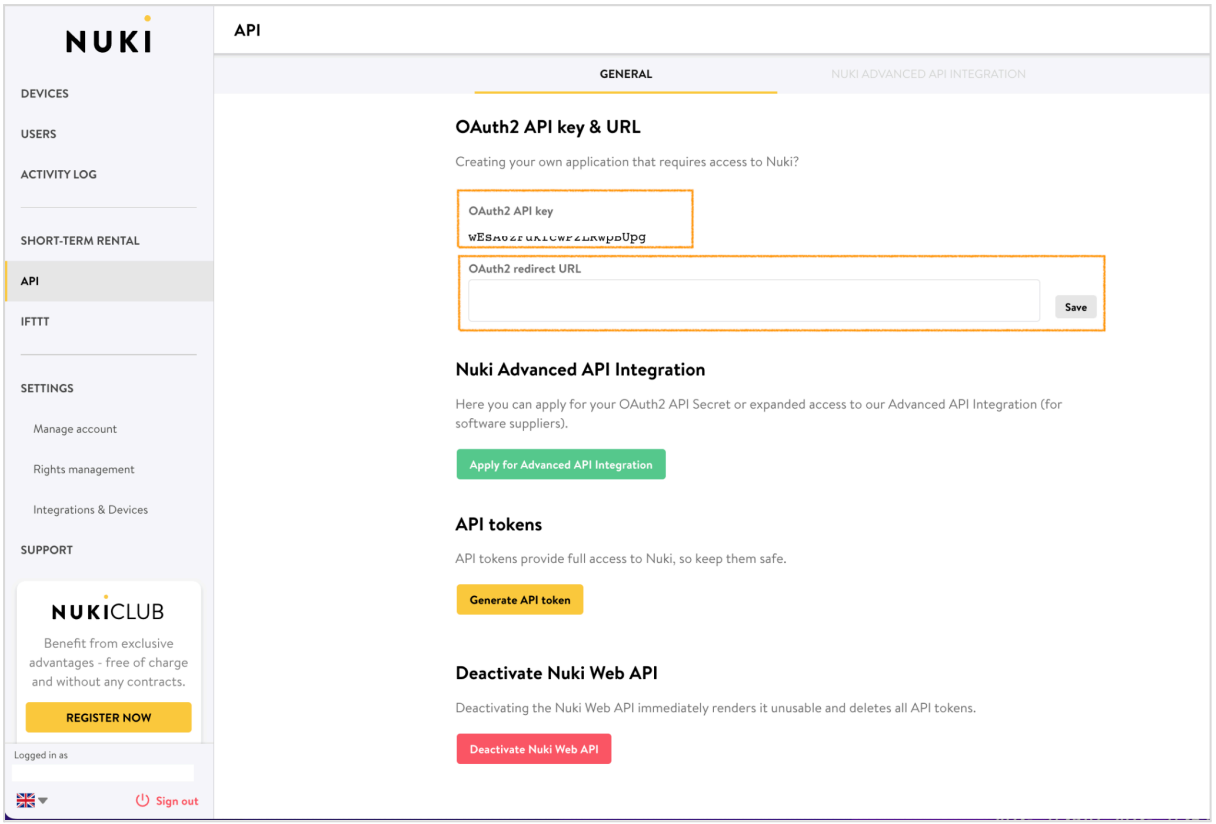
Parameter	Description	Value
response_type	The type of grant type that is used.	value = "code" as Nuki Web uses the authorization code flow

client_id	The ID of the client that is making the request.	CLIENT_ID = OAuth2 API Key from Nuki Web > Menu > API (refer to the image below)
redirect_uri	The callback URL to which the users will be redirected after they successfully logged in. This field is mandatory and has to be set in Nuki Web. You can add several URIs as comma separated values, but it is not allowed to contain wildcard parameters.	CALLBACK_URL = OAuth2 redirect URL from Nuki Web > Menu > API (refer to the image below)
scope	List of scopes that you want to request from the user for your application.	For example, SCOPES = account smartlock smartlock.auth smartlock.config (refer to the Definitions Section J: Scopes)
state (optional)	A random value that is generated by the requesting application to prevent cross-site request forgery (CSRF) attacks. If it is not provided with the request, the server will generate one.	For example, state = "KJHg876HJHjklj9876HJkkl7sdf"

NOTE: All parameters need to be URL encoded in all the requests during the flow. Refer to the Online URL encoder/decoder if needed.

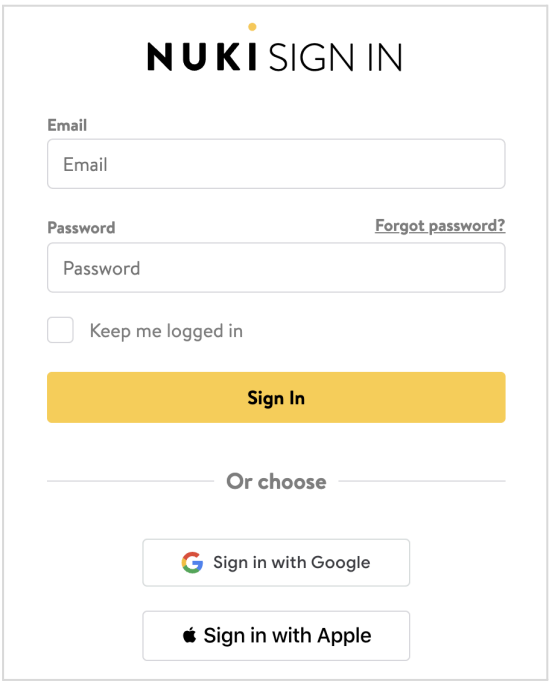
Example Authorization Call:

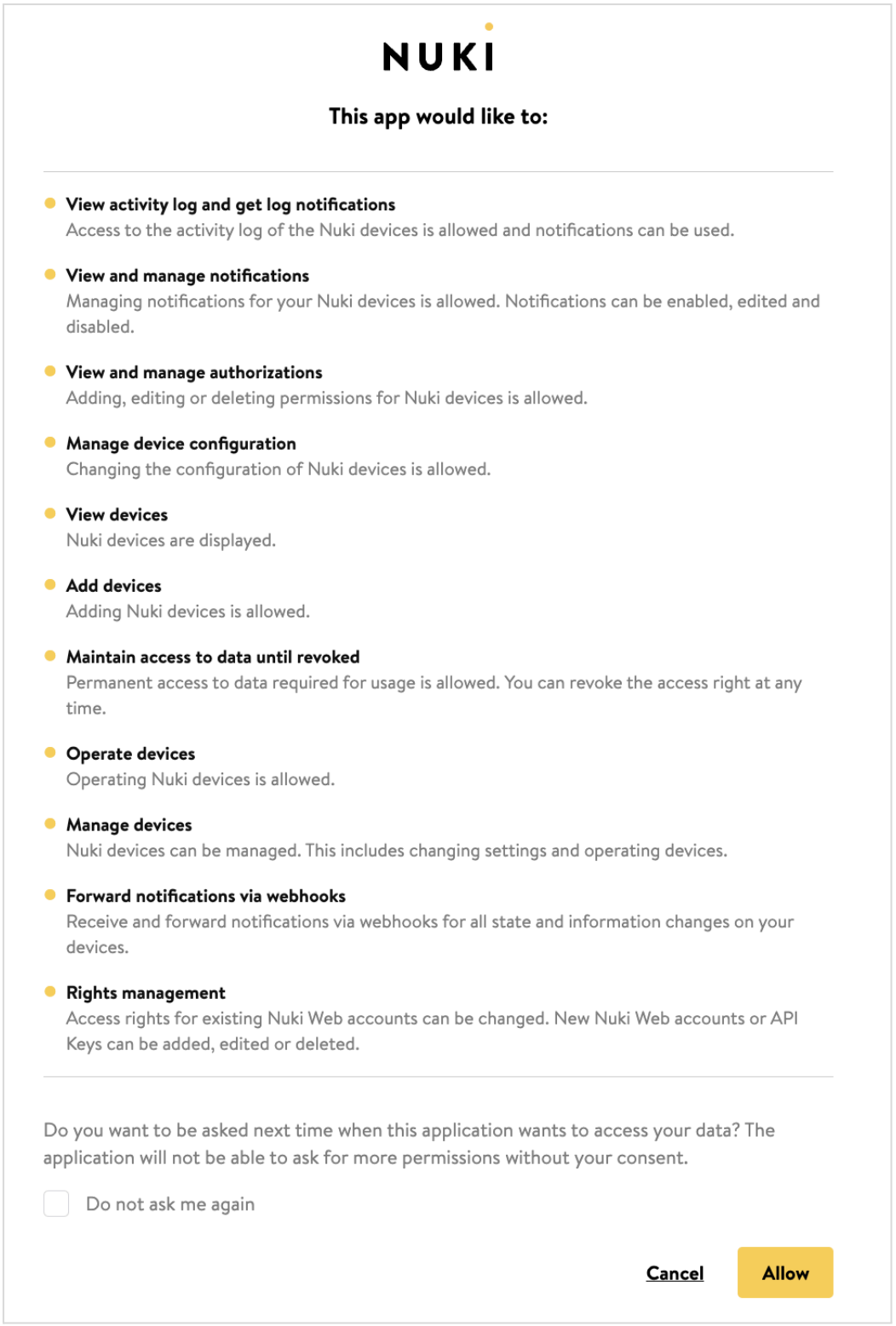
```
https://api.nuki.io/oauth/authorize?response_type=code&client_id=v7kn_NX7vQ7VjQdXFGK43g&redirect_uri=https%3A%2F%2Ftest.com&scope=account%20smartlock%20smartlock.create%20smartlock.auth%20smartlock.action%20smartlock.log%20smartlock.readOnly%20smartlock.config%20notification
```



1.2 User authorizes the client application

When the user clicks on the link or button within the client application, they should authorize by entering their Nuki Web username and password and accepting the scopes mentioned.





1.3 Client application receives the authorization code

Upon successful authorization, the user will be redirected to the provided callback URL (redirect URI) mentioned in 1.1, which contains the authorization code identified by “code”.

For example,

```
https://test.com/?code=LN6LcsYquiZG2Zbp4hqpRpbgguyFONJvDBtTvQQirSw.bz5xJmD4_SR4rzajiefv3kTlD4CfvRd55rjmwH7T7xM&scope=account+notification+smartlock+smartlock.readOnly+smartlock.action+smartlock.auth+smartlock.config+smartlock.log+offline_access&state=7eLK8cFTsqexBur4LBEFLAgMyZB5c8Hj
```

AUTHORIZATION_CODE = LN6LcsYquiZG2Zbp4hqpRpbgguyFONJvDBtTvQQirSw.bz5xJmD4_SR4rzajiefv3kTlD4CfvRd55rjmwH7T7xM

1.4 Client application receives the access token

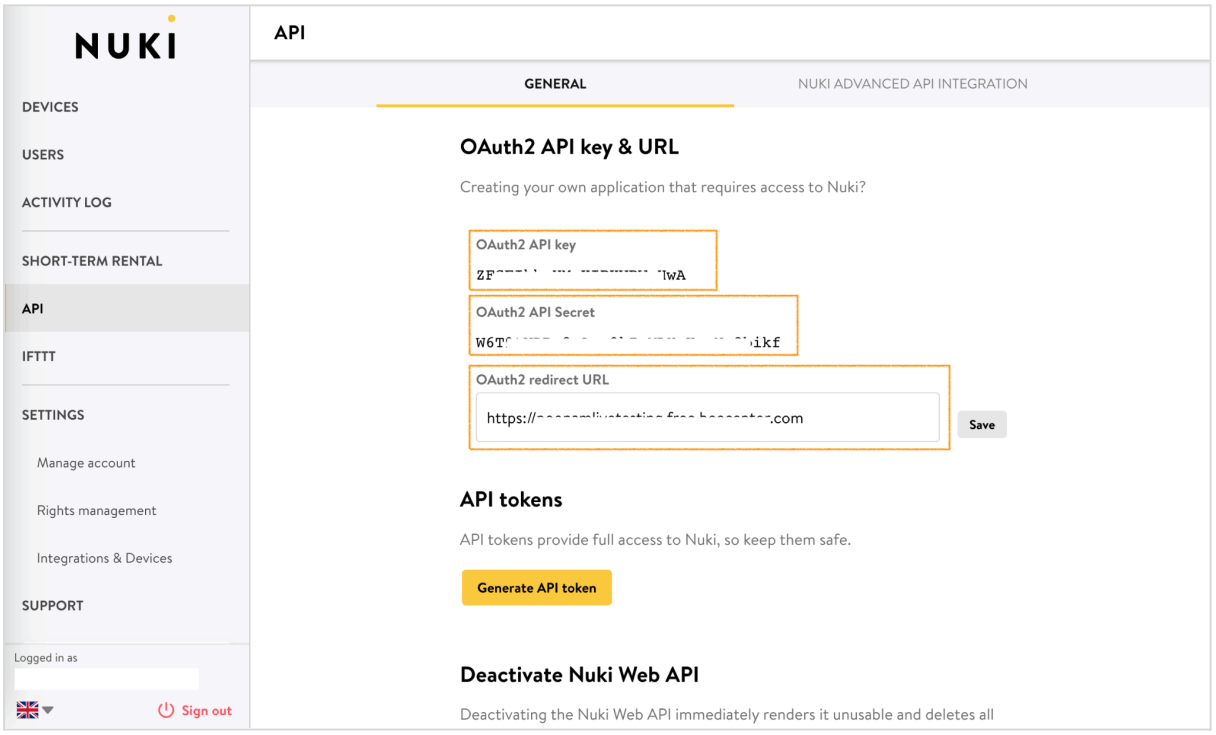
The client application posts to the /oauth/token URL to exchange the authorization code obtained in 1.3 to receive the access token. This access token can be used to access the user's Nuki Web account data.

Below is the curl command to request the access token:

```
curl -X POST -d 'client_id=CLIENT_ID client_secret=CLIENT_SECRET grant_type=authorization_code code=AUTHORIZATION_CODE redirect_uri=CALLBACK_URL' https://api.nuki.io/oauth/token
```

Parameter	Description	Value
client_id	The ID of the client that is making the request.	CLIENT_ID = OAuth2 API Key from Nuki Web > Menu > API (refer to the image below)
client_secret	The secret key of the client that is making the request (only available after the advanced API request is approved).	CLIENT_SECRET = OAuth2 API Secret from Nuki Web > Menu > API (refer to the image below)
code	The authorization code returned in the redirect URL upon successful user authorization.	AUTHORIZATION_CODE obtained from the redirect URL as mentioned in 1.3
redirect_uri	The callback URL to which the users will be redirected after they successfully logged in. This field is mandatory and has to be set in Nuki Web. You can add several URIs as comma separated values, but it is not	CALLBACK_URL = OAuth2 redirect URL from Nuki Web > Menu > API (refer to the image below)

	allowed to contain wildcard parameters.	



For example, the request can be:

```
curl --location 'https://api.nuki.io/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode
'code=LN6LcsYquiZG2Zbp4hqpRpbgguyFONJvDBtTvQQirSw.bz5xJmD4_SR4rzaj
iefv3kTlD4CfvRd55rjmwH7T7xM' \
--data-urlencode 'scope=account smartlock smartlock.create
smartlock.auth smartlock.action smartlock.log smartlock.readOnly
smartlock.config notification' \
--data-urlencode 'client_id=v7kn_NX7vQ7VjQdXFGK43g' \
--data-urlencode 'client_secret=Mn2XAJ5A0hWfcF2BX3xo3G32UIdyaTe1'
\
--data-urlencode 'redirect_uri=https://www.test.com'
```

For example, the response from the authorization server will be:

```
{
    "access_token":
"fCrFkSDhnXtO3YHeYog_jT6AwXxNtt3vKIDW-v9W4Go.tlN3vFYsaleGiGFbvxJdP
VKqAcDPzQF36EKRqQmH8K0",
    "token_type": "bearer",
    "expires_in": 3600,
```

```
    "refresh_token":
"RTQVsMYDk6eedwmY05_unczUIHPNxJOKsRyne4Kf_KY.RiCNJRQlv7XF4zQxk7dbw
QLMlDwfTlmD-EN2bOGQzNr",
    "scope": "account notification smartlock smartlock.readOnly
smartlock.action smartlock.auth smartlock.config smartlock.log
offline_access"
}
```

The result is an access token that is valid for **one hour** (3600 seconds). The obtained ACCESS_TOKEN can be used to make requests to the Nuki Web API.

For example,

```
curl -X GET --header 'Accept: application/json' --header
'Authorization: Bearer ACCESS_TOKEN'
'https://api.nuki.io/smartlock'
```

Since the access token expires in one hour, the authorization server also provides a refresh token to renew the access token. The refresh token is valid for 90 days.

1.5 Client application uses the refresh token to renew the access token

After the access token expires, the API will provide an “Invalid Token Error”. Thus, you can use the REFRESH_TOKEN obtained in 1.4 to get a new access token with the following URL:

```
curl -X POST -d
“grant_type=refresh_token&client_id=CLIENT_ID&client_secret=CLIENT
_SECRET&refresh_token=REFRESH_TOKEN”
https://api.nuki.io/oauth/token
```

For example, the request can be:

```
curl --location 'https://api.nuki.io/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
--data-urlencode 'client_id=v7kn_NX7vQ7VjQdXFGK43g' \
--data-urlencode 'client_secret=Mn2XAJ5A0hWfcF2BX3xo3G32UIdyaTe1'
\
--data-urlencode
'refresh_token=RTQVsMYDk6eedwmY05_unczUIHPNxJOKsRyne4Kf_KY.RiCNJRQ
lv7XF4zQxk7dbwQLMlDwfTlmD-EN2bOGQzNr'
```

For example, the response from the authorization server will be:

```
{
    "access_token":
"NCIrFmY4c0yRaqKLsagNtOKO11GZdb8B5qhSuGnVaQEWc.hTMpRm6bsUZBI07XMLj
NPYSDz-J5_aW3sd59BTITZx1",
    "token_type": "bearer",
```

```
    "expires_in": 3600,
    "refresh_token":
"4rtF2YVJo1CLW1mQrydksl47ZSGnY2rJbzkKBcWEjIXo.m41NMBwQleetajGfrdEl
w3_gnBM2ldo-VzK-B269pbW",
    "scope": "account notification smartlock smartlock.readOnly
smartlock.action smartlock.auth smartlock.config smartlock.log
offline_access"
}
```

This refresh token can be used to use the user's data securely. It is valid for 90 days, after which it is regenerated when you request a new access token.

2. API Token

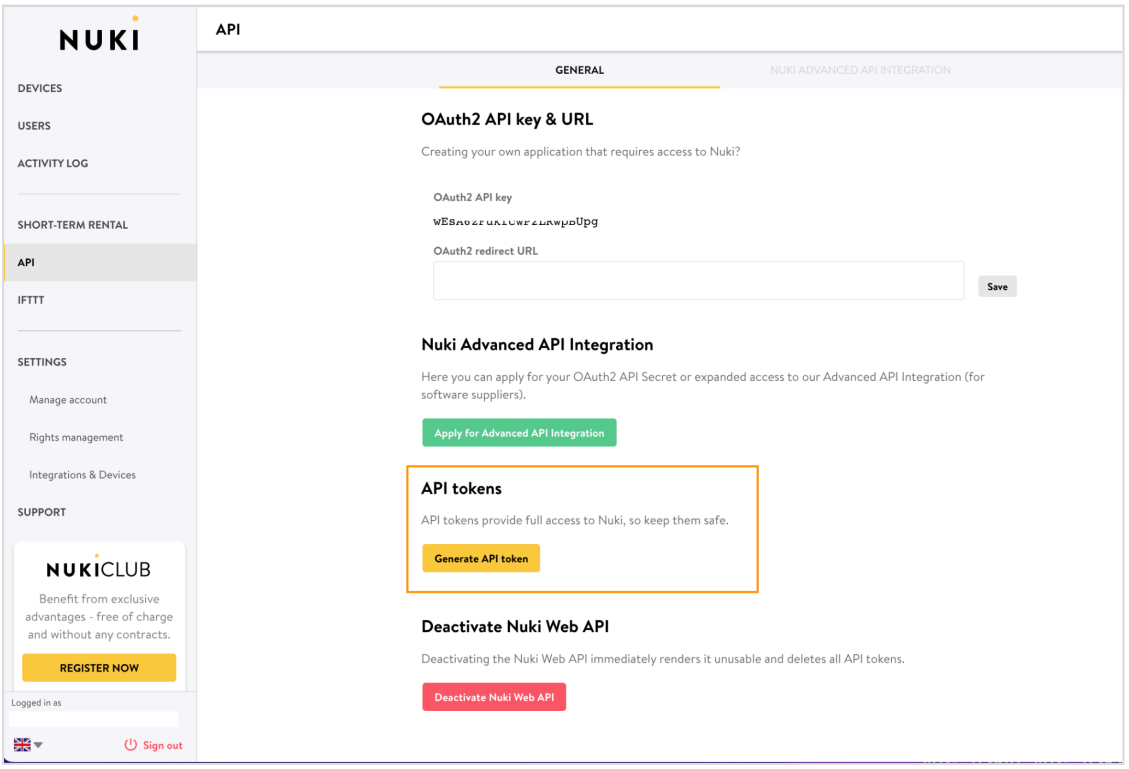
When you login to Nuki Web and access the API section, the "API tokens" feature is used to create API tokens.

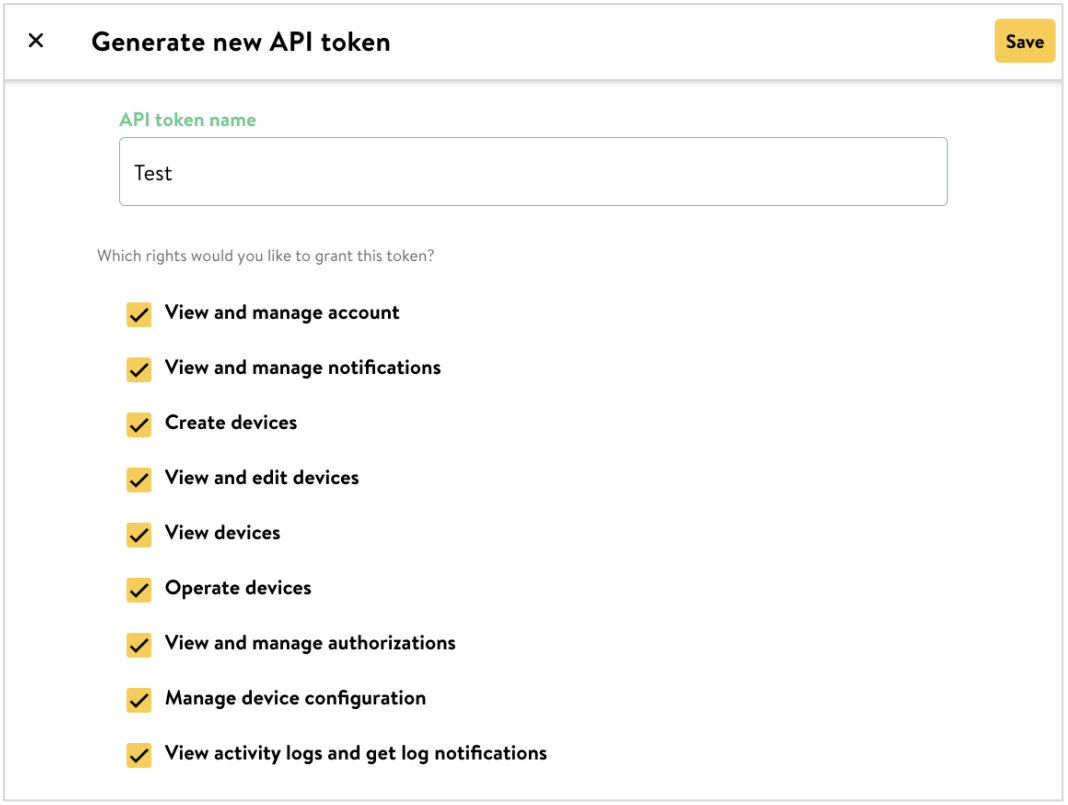
Click on "Generate API token" to create one. Copy the API token into the clipboard and store it in a secure way. It gives permanent access to all rights you granted while creating the token.

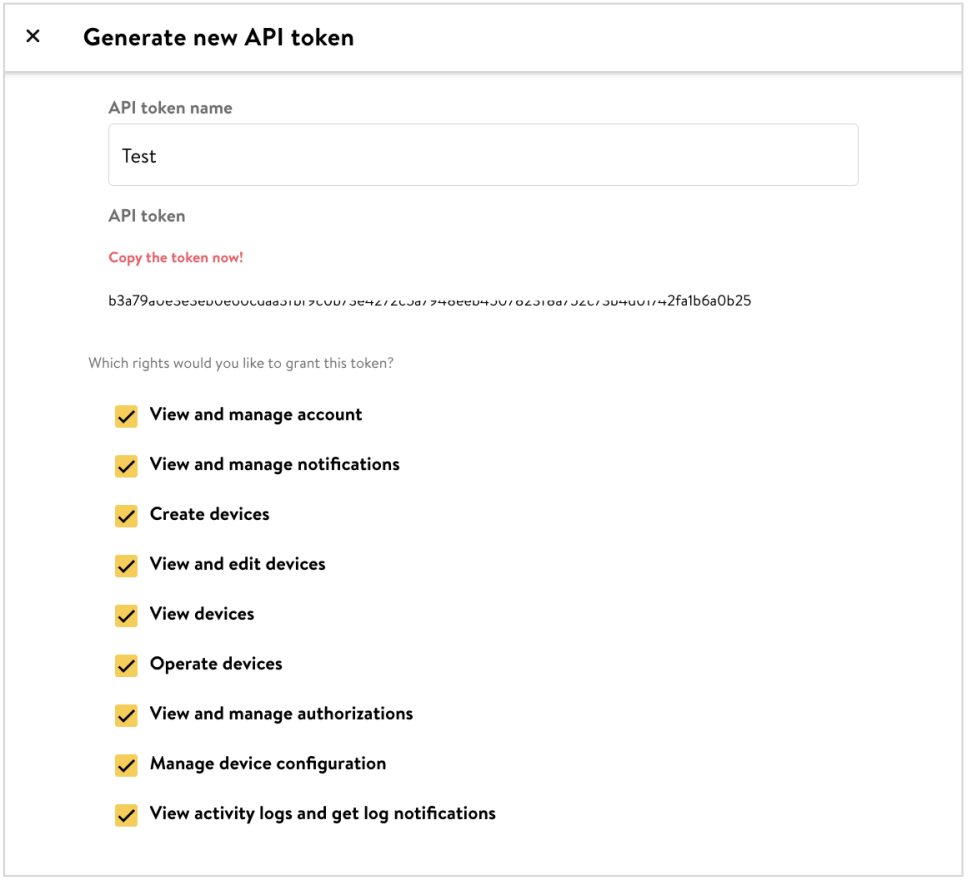
Use it as the "Authorization: Bearer" in your API calls:

```
curl -X GET --header 'Accept: application/json' --header
'Authorization: Bearer API_token' 'https://api.nuki.io/smartlock'
```

API tokens do not expire, but they are **destroyed when the password** of the corresponding Nuki Web account **changes**.




Generate new API token
Save
API token name
Test
Which rights would you like to grant this token?
View and manage account
View and manage notifications
Create devices
View and edit devices
View devices
Operate devices
View and manage authorizations
Manage device configuration
View activity logs and get log notifications


Generate new API token
API token name
Test
API token
Copy the token now!
Which rights would you like to grant this token?
View and manage account
View and manage notifications
Create devices
View and edit devices
View devices
Operate devices
View and manage authorizations
Manage device configuration
View activity logs and get log notifications

6. Configure webhooks with Advanced API access

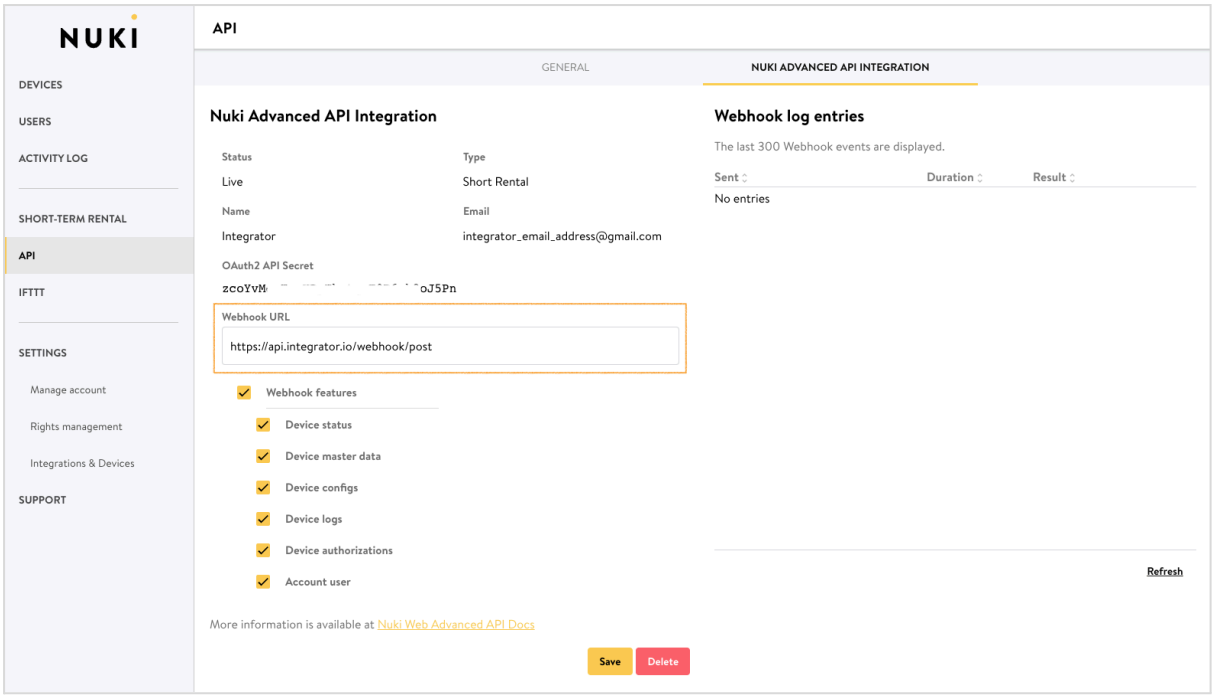
This section is relevant for integrators who want to receive webhooks. The prerequisite for configuring webhooks is to apply for the Nuki Web Advanced API access as explained in the Section 4: Apply for Nuki Advanced API access.

An integrator can choose between two workflows (explained in the Core Concepts Section D: Central and Decentral Webhooks):

- Central - the integrator has a single distinct URL
- Decentral - the integrator has several different URLs

6.1. Steps to set up central webhooks:

- Upon getting the Nuki Advanced API request approved by the Nuki team, the integrator can set up the central URL in the “Nuki Advanced API Integration” tab in the API section of Nuki Web.
- The integrator must enable the “Webhook features to receive device-triggered webhooks.



- The integrator needs to perform an authorization with OAuth2 Code flow, with the scope “**webhook.central**”. Other scopes are optional, but without the webhook.central scope, webhooks are not receivable.
- For example, the request to obtain the authorization code is:

```
https://api.nuki.io/oauth/authorize?response_type=code&client_id=v7kn_NX7vQ7VjQdXFGK43g&redirect_uri=https%3A%2F%2Ftest.com&scope=account%20smartlock%20smartlock.create%20smartlock.auth%20smartlock.action%20smartlock.log%20smartlock.readOnly%20smartlock.config%20notification%20webhook.central
```

- Upon successful authentication, an **authorization code** is returned to the specified redirect URI, which can be used to obtain an access code.
- To obtain an access token, call the POST request as shown in the example below:

```
curl --location 'https://api.nuki.io/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode
'code=LN6LcsYquiZG2Zbp4hqpRpbgguyFONJvDBtTvQQirSw.bz5xJmD4_SR
4rzajiefv3kTlD4CfvRd55rjmwH7T7xM' \
--data-urlencode 'scope=account smartlock smartlock.create
smartlock.auth smartlock.action smartlock.log
smartlock.readOnly smartlock.config notification
webhook.central
' \
--data-urlencode 'client_id=v7kn_NX7vQ7VjQdXFGK43g' \
--data-urlencode
'client_secret=Mn2XAJ5A0hWfcF2BX3xo3G32UIdyaTe1' \
--data-urlencode 'redirect_uri=https://www.test.com'
```

- A returned access token ensures a valid central webhook registration. This concludes the central webhook registration process. The access token is usable for further actions using the Nuki Web API.

NOTE: Every central webhook POST contains a header field "X-Nuki-Signature-SHA256" with the signature value of the signed body's payload. The JSON body is signed with the HMAC SHA256 algorithm based on RFC2104, with the "Client Secret" as the signing key.

6.2. Steps to set up decentral webhooks:

- The integrator must ensure that the “Nuki Advanced API Integration” tab is active in the API section of Nuki Web.
- The configured webhook url and webhook features are irrelevant, because for decentral webhooks specific configuration steps are needed.
- The integrator needs to perform an authorization with OAuth2 Code flow, with the scope “**webhook.decentral**”. Other scopes are optional, but without the webhook.decentral scope, webhooks are not receivable.
- For example, the request to obtain the authorization code is:

```
https://api.nuki.io/oauth/authorize?response_type=code&client
_id=v7kn_NX7vQ7VjQdXFGK43g&redirect_uri=https%3A%2F%2Ftest.co
m&scope=account%20smartlock%20smartlock.create%20smartlock.au
th%20smartlock.action%20smartlock.log%20smartlock.readOnly%20
smartlock.config%20notification%20webhook.decentral
```

- Upon successful authentication, an **authorization code** is returned to the specified redirect URI, which can be used to obtain an access code.

- To obtain an access token, call the POST request as shown in the example below:

```
curl --location 'https://api.nuki.io/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode
'code=LN6LcsYquiZG2Zbp4hqpRpbgguyFONJvDBtTvQQirSw.bz5xJmD4_SR
4rzajiefv3kTlD4CfvRd55rjmwH7T7xM' \
--data-urlencode 'scope=account smartlock smartlock.create
smartlock.auth smartlock.action smartlock.log
smartlock.readOnly smartlock.config notification
webhook.decentral
' \
--data-urlencode 'client_id=v7kn_NX7vQ7VjQdXFGK43g' \
--data-urlencode
'client_secret=Mn2XAJ5A0hWfcF2BX3xo3G32UIdyaTe1' \
--data-urlencode 'redirect_uri=https://www.test.com'
```

- To register a decentral webhook with this access token, place a PUT request to:

```
PUT https://api.nuki.io/api/decentralWebhook/
```

For example, the curl command is:

```
curl --location --request PUT
'https://api.nuki.io/api/decentralWebhook' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
fCrFkSDhnXtO3YHeYog_jT6AwXxNtt3vKIDW-v9W4Go.tlN3vFYsaleGiGFbv
xJdPVKqAcDPzQF36EKRqQmH8K0' \
--data '{
    "webhookFeatures": [
        "DEVICE_STATUS",
        "DEVICE_MASTERDATA",
        "DEVICE_CONFIG",
        "DEVICE_LOGS",
        "DEVICE_AUTHS",
        "ACCOUNT_USER"
    ],
    "webhookUrl": "https://decentral123.webhook.at"
```

- The response contains the secret and the identifier of the registered webhook.

```
{
    "id": 54294256,
    "secret": "4aCV5rT556edR6zn8VBniq78PbZlfL8R",
    "webhookUrl": "https://decentral123.webhook.at",
    "webhookFeatures": [
        "DEVICE_STATUS",
        "DEVICE_MASTERDATA",
```

```
        "DEVICE_CONFIG",
        "DEVICE_LOGS",
        "DEVICE_AUTHS",
        "ACCOUNT_USER"
    ]
}
```

- If the PUT request to https://api.nuki.io/api/decentralWebhook is successful, decentral webhooks are enabled. This concludes the decentral webhook registration process.

- To unregister a decentral webhook a DELETE request has to be made to:
  `DELETE https://api.nuki.io/api/decentralWebhook/{id}`

  For example, the curl command is:
  ```
  curl --location --request DELETE
  'https://api.nuki.io/api/decentralWebhook/54294256' \
  --header 'Authorization: Bearer
  fCrFkSDhnXtO3YHeYog_jT6AwXxNtt3vKIDW-v9W4Go.tlN3vFYsaleGiGFbv
  xJdPVKqAcDPzQF36EKRqQmH8K0'
  ```

- All registered decentral webhooks can be obtained from:
  `GET https://api.nuki.io/api/decentralWebhook/`

  For example, the curl command is:
  ```
  curl --location 'https://api.nuki.io/api/decentralWebhook' \
  --header 'Authorization: Bearer
  fCrFkSDhnXtO3YHeYog_jT6AwXxNtt3vKIDW-v9W4Go.tlN3vFYsaleGiGFbv
  xJdPVKqAcDPzQF36EKRqQmH8K0'
  ```

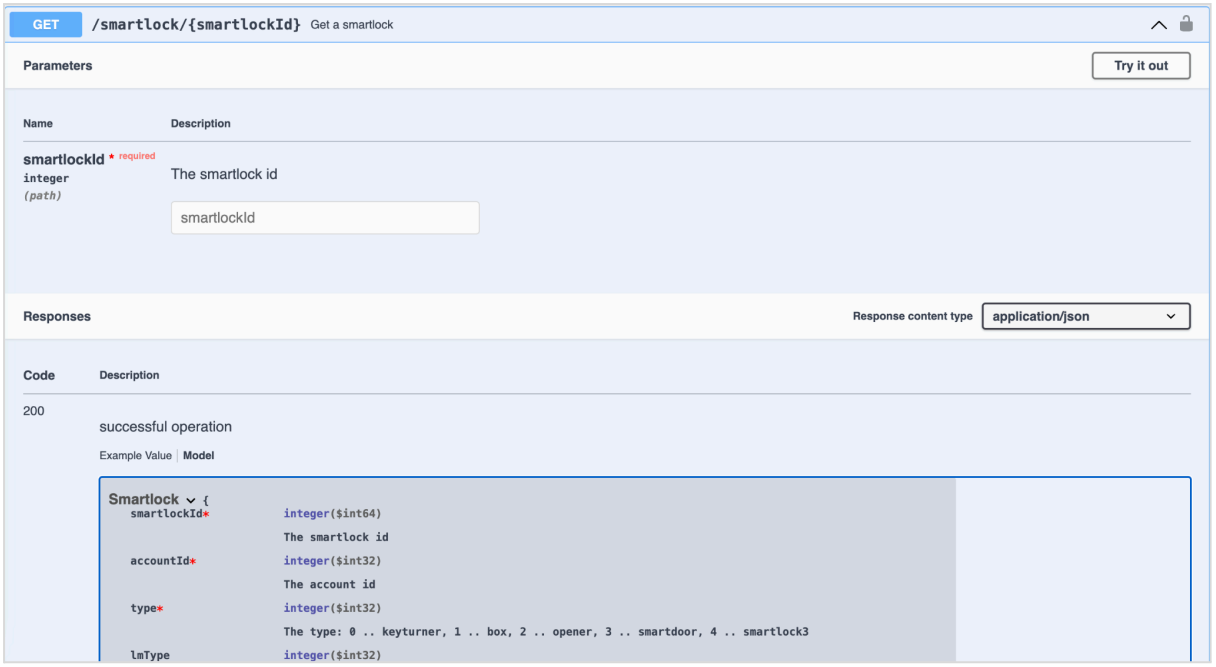
Try the demo - Your first API call on Swagger

The Nuki Web API can be found under the calling URL https://api.nuki.io.

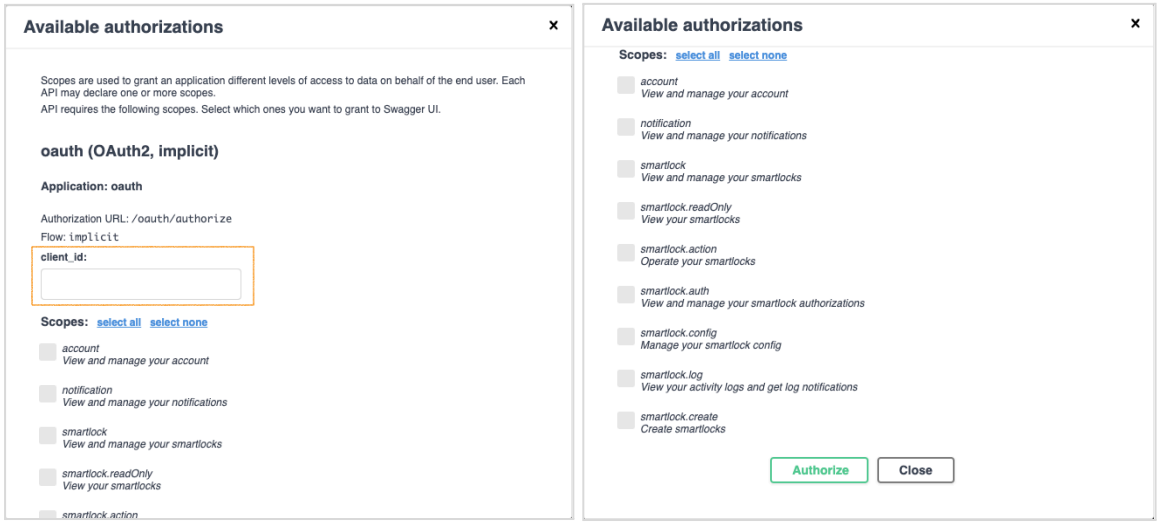
Swagger is a tool we use to automatically generate documentation from our OpenAPI definition for visual interaction and easier testing for you. You can find the link to Swagger in the Developer Forum.

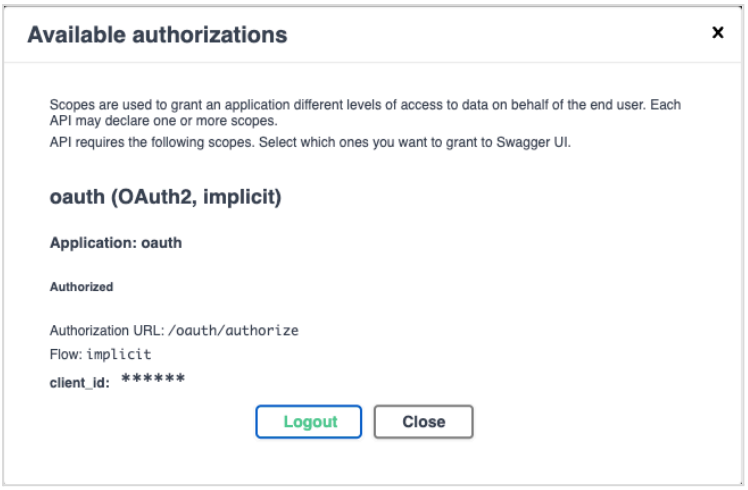
The Swagger interface lists all API commands with its input and output parameters. It also allows you to easily try out the API commands from the interface itself.

There is a section called “**Models**” which describes all the parameters in detail. Each endpoint also has the model details to explain the parameters in detail. A sample model looks like this:



In order to make an API call from the Swagger interface, **first authorize it with a client_id** (i.e. client_id = OAuth2 API Key from your Nuki Web > Menu > API). Select the scopes that you would like to, and authorize.



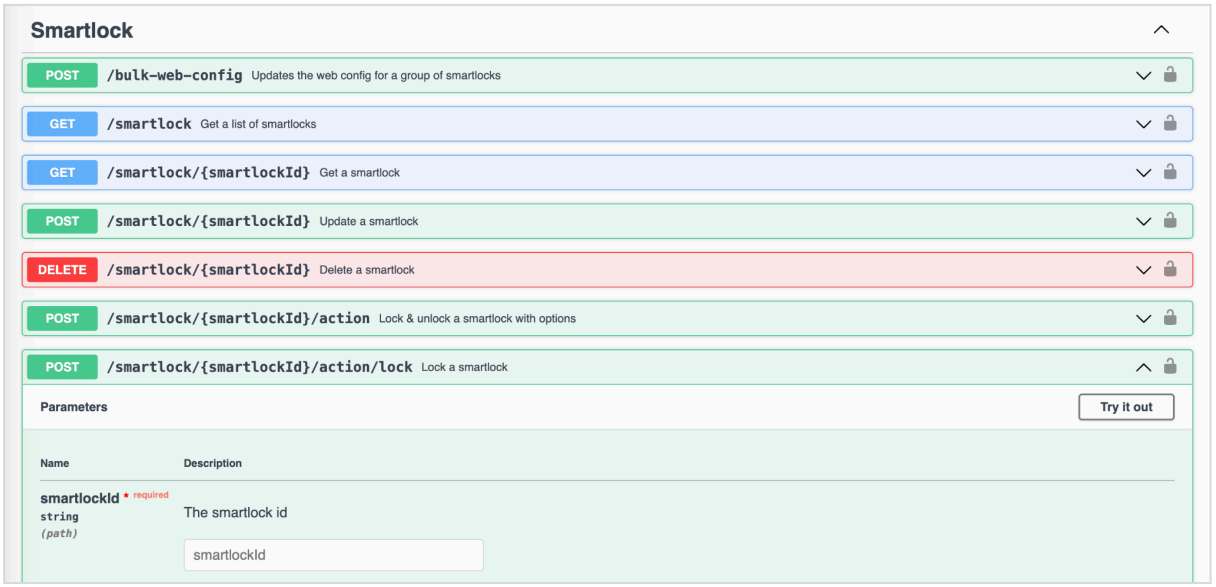


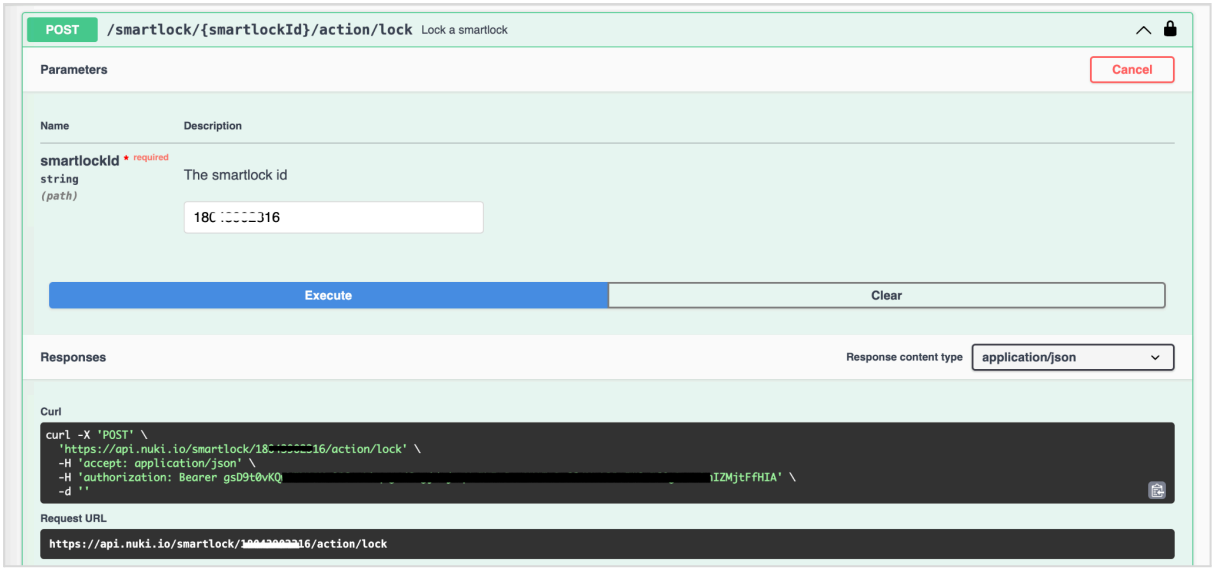
Once it is authorized, you can try out any endpoint.

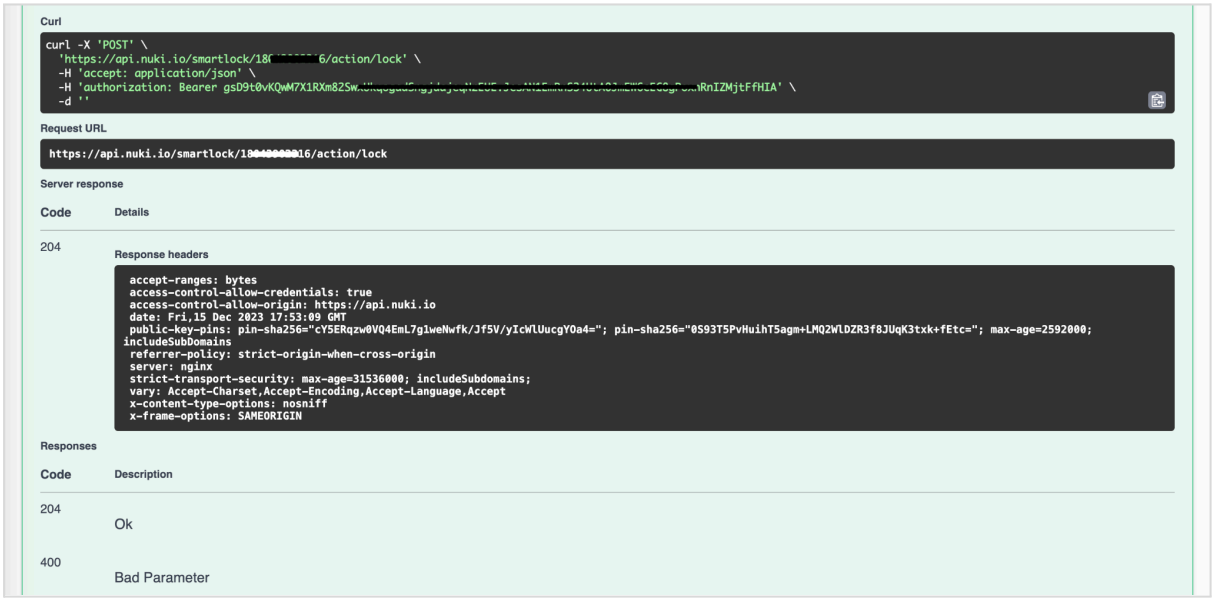
NOTE: Advanced API endpoints can only be used if you have an Advanced API request approved.

For your first API call, you could try to lock your Smart Lock via the API.

- Under the Smartlock section, select the POST endpoint to lock a smart lock.
- Click on the “Try it out” button which allows you to enter the smartlockId (in decimal, refer to the conversion from HEX to DEC here).
- Click “Execute” and see the response in the interface.
- You will also get the corresponding cURL call along with the response from the API.







Congratulations! You've executed your first API call.

You can also download a configuration file for the API at https://api.nuki.io/static/swagger/swagger.json or directly open the file in Postman (File > Import > Link) to create a Postman Collection for it.

FEATURES

This document lists the most commonly used features with Nuki Web API. In order to try out the features, you must meet the below preconditions:

- You should obtain the authorization token through an appropriate method (refer to Section 5: Authenticate calls to the Nuki Web API)
- Your device must be online
- You must convert your device ID {smartlockId} from Hex to Decimal (refer to Core Concepts Section A: Convert Device ID from Hex to Decimal)
    - (or) call the GET /smartlock/ endpoint to get the SL ID from the API

1. Check if a device is online

The Nuki Web API commands will work only if a device is online. You can check the "serverState" of the device to check if it is online or offline.

- If serverState = 0, it means the SL is online
- If serverState = 4, it means the SL is offline

API endpoint	GET /smartlock/{smartlockId}

curl command	`curl -X 'GET' \`
	`  'https://api.nuki.io/smartlock/SMARTLOCK_ID \`
	`  -H 'accept: application/json' \`
	`  -H 'authorization: Bearer ACCESS_TOKEN'`

With the Advanced API access, you can also receive webhooks for lock state changes and server state by selecting the "Device status" or "Device master data" Webook features in the Nuki Advanced API Integration section of Nuki Web.

Webhook response

```
{
  "headers": {
    "X-Nuki-Signature-SHA256": "xxx",
    "X-Nuki-Signature": "xxx",
    "Content-Type": "application/json;
charset=UTF-8"
  },
  "body": {
    "feature": "DEVICE_STATUS",
    "smartlockId": SMARTLOCK_ID,
    "state": {
      "mode": 2,
      "state": 3,
      "trigger": 0,
      "lastAction": 1,
      "batteryCritical": false,
      "batteryCharging": false,
      "batteryCharge": 32,
```

```
      "keypadBatteryCritical": false,
      "doorsensorBatteryCritical": false,
      "doorState": 0,
      "ringToOpenTimer": 0,
      "nightMode": false
    },
    "serverState": 0,
    "adminPinState": 0
  },
  "timestamp": "2024-01-05T09:05:14.200Z",
  "path": "WEBHOOK_URL"
}
```


NOTE: If serverState = 1/2/3, the connection of the device to the Nuki Web account needs to be reestablished.

2. Lock & unlock a door

Ensure to set the door handle type correctly for your door. You can do this in the Nuki App: Features & Configuration > General > Choose door fitting (Knob, Lever or Lift-up Handle)
For more information on this, refer to Definitions Section L: Smart Lock actions.

To lock your door, you can directly call the lock command

API endpoint	POST /smartlock/{smartlockId}/action/lock
curl command	`curl -X 'POST' \ 'https://api.nuki.io/smartlock/SMARTLOCK_ID/action/lock' \ -H 'accept: application/json' \ -H 'authorization: Bearer ACCESS_TOKEN' \ -d ''`

To open your door with handle type Knob, you can directly call the unlock command

API endpoint	POST /smartlock/{smartlockId}/action/unlock
curl command	`curl -X 'POST' \ 'https://api.nuki.io/smartlock/SMARTLOCK_ID/action/unlock' \ -H 'accept: application/json' \ -H 'authorization: Bearer ACCESS_TOKEN' \ -d ''`

For the door with handle type Lever, the above command only unlocks but doesn't open the door (i.e. unlatch).

To unlock but not unlatch the door with handle type Knob, you can send an action = 1 (applicable for Smart Locks only)

API endpoint	POST /smartlock/{smartlockId}/action/
	{
	“action”: 1
	}

curl command	`curl -X 'POST' \`
	`'https://api.nuki.io/smartlock/SMARTLOCK_ID/action' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"action": 1`
	`}'`

If you would like to unlock and unlatch (i.e. open the door), you can send an action = 3

API endpoint	POST /smartlock/{smartlockId}/action/
	{
	“action”: 3
	}

curl command	`curl -X 'POST' \`
	`'https://api.nuki.io/smartlock/SMARTLOCK_ID/action' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"action": 3`
	`}'`

3. Invite a user

If you would like to provide access permissions to another user (family or friend) for your devices, you can invite them to use the Nuki App. You can create a user with their email address.

Firstly, create an account user

API endpoint	PUT /account/user { "email": "john_doe@gmail.com",

	"name": "john_doe", "language": "en" //*Available: en, de, es, fr, it, nl, cs, sk* }

curl command	`curl -X 'PUT' \`
	`'https://api.nuki.io/account/user' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"email": "john_doe@gmail.com",`
	`"name": "john_doe",`
	`"language": "en"`
	`}'`

Then create an auth and assign the devices (one or more) by providing the accoundUserId received in the response of the previous call

API endpoint	PUT /smartlock/auth
	{
	"name": "john_doe",
	"accountUserId": "1264915785", //for the user created above
	"type": "0",
	"smartlockIds": [
	"18891899123",
	"18891899124"
	]
	}

curl command	`curl -X 'PUT' \`
	`'https://api.nuki.io/smartlock/auth' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"name": "john_doe",`
	`"accountUserId": 1264915785,`
	`"smartlockIds": [`
	`"18891899123",`
	`"18891899124"`
	`],`
	`"type": 0`
	`}'`

The user receives an email with an invitation code that needs to be redeemed in the Nuki App within 48 hours.

4. Create a keypad code

Firstly, ensure a Keypad is paired with the Smart Lock. It is available in the “config” of a Smart Lock as keypadPaired = true. The keypad code is a 6-digit code. It cannot contain “0” and must not start with “12”. The keypad code is also unique per device.

To create a keypad code, auth type = 13

API endpoint	PUT /smartlock/auth
	{
	"name": "test_keypad_code",
	"type": "13",
	"code": "252525",
	"smartlockIds": [
	"18891899123"
	]
	}

curl command	`curl -X 'PUT' \`
	`'https://api.nuki.io/smartlock/auth' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"name": "test_keypad_code",`
	`"smartlockIds": [`
	`"18891899123"`
	`],`
	`"type": 13,`
	`"code": 252525`
	`}'`

5. Check the battery status

To check the battery status of a device from the API, call GET /smartlock/{smartlockId} and look under the “state” section for the “batteryCharge” parameter. The number indicates the current % of battery or the remaining capacity of the battery in %.

6. Get the activity logs

To get the activity logs for a device, call GET /smartlock/{smartlockId}/log. You can get only a maximum of up to 50 logs from the Nuki Web API, even though the logs up to 3 months are stored and visible in Nuki Web for standard users.

With the Advanced API access, you can also receive webhooks for new activity log entries by selecting the “Device Status” Webook feature in the Nuki Advanced API Integration section of Nuki Web.

Interpretation of the logs:

Parameter	Values
deviceType	0... smartlock 1/2, 1... box, 2... opener, 3... smartdoor, 4... smartlock 3/4, 5... smartlock 5
action	1 .. unlock, 2 .. lock, 3 .. unlatch, 4 .. lock'n'go, 5 .. lock'n'go with unlatch, 208 .. door warning ajar, 209 door warning status mismatch, 224 .. doorbell recognition (only Opener), 240 .. door opened, 241 .. door closed, 242 .. door sensor jammed, 243 .. firmware update, 250 .. door log enabled, 251 .. door log disabled, 252 .. initialization, 253 .. calibration, 254 .. (activity) log enabled, 255 .. (activity) log disabled
trigger	0 .. system, 1 .. manual, 2 .. button, 3 .. automatic, 4 .. web, 5 .. app, 6 .. auto lock, 7 .. accessory, 255 .. keypad
state	0 .. Success, 1 .. Motor blocked, 2 .. Cancelled, 3 .. Too recent, 4 .. Busy, 5 .. Low motor voltage, 6 .. Clutch failure, 7 .. Motor power failure, 8 .. Incomplete, 9 .. Rejected, 10 .. Rejected night mode, 254 .. Other errors, 255 .. Unknown error
autoUnlock	True if it was an auto unlock
source	1 .. Keypad code, 2 .. Fingerprint, 0 .. Default
error	In case of any error, it contains the error message

Short Rental Use Cases

The Short Rental API has the same base URL as the Nuki Web API. Note that the Nuki Smart Hosting subscription is necessary for using Smart Locks for short rental purposes. Integrators who would like to use the API for short rental purposes are required to use the Advanced API , by applying for an OAuth2 authorization token under the “Short Rental” category (refer to Section 5: Authenticate calls to Nuki Web API).

Additionally, the Smart Locks must have an active Smart Hosting subscription. If the devices do not have an active Smart Hosting subscription, the API (v.1.5.0. onwards) will return an error message, as documented here.

7. Check if a device has Smart Hosting

To check if a device (Smart Lock or Smart Door) has an active Smart Hosting subscription, call GET /smartlock/{smartlockId}.

- If the device has an active Smart Hosting subscription, the response contains the subscription details:
  "currentSubscription": {
  "type": "B2C",
  "creationDate": "2023-08-11T08:14:11.981Z"
  }
- If the device doesn't have an active Smart Hosting subscription, the response contains an "error": "No active Smart Hosting subscription for the Smart Lock, please get one!"

8. Create guest permissions with a limited access time

It is a good practice to create a user for every new booking and then create auths for all guests in the booking. This ensures that the guests can access the vacation home either through a keypad code or by using the Nuki App.

8.1 Create a guest user indicated by the booking number

API endpoint	PUT /account/user
	{
	"email": "guest_email@gmail.com",
	"name": "BOOKING123",
	"language": "en" *//Available: en, de, es, fr, it, nl, cs, sk, pl*
	}

curl command	`curl -X 'PUT' \`
	`'https://api.nuki.io/account/user' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"email": "guest_email@gmail.com",`
	`"name": "BOOKING123",`
	`"language": "en"`
	`}'`

8.2 Create auths and assign the devices by providing the accoundUserId received in the response of the previous call; booking from 20-Dec-2023 1PM to 25-Dec-2023 11AM

API endpoint	PUT /smartlock/auth { "name": "BOOKING123 #1", // for guest 1

	"accountUserId": "1264915785", // for the guest user created above "type": "0", "smartlockIds": ["18891899123", "18891899124"], "remoteAllowed": false, "allowedFromDate": "2023-12-20T13:00:00.000Z", "allowedUntilDate": "2023-12-25T11:00:00.000Z", "allowedWeekDays": 127 }

curl command	`curl -X 'PUT' \`
	`'https://api.nuki.io/smartlock/auth' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"name": "BOOKING123 #1",`
	`"accountUserId": 1264915785,`
	`"allowedFromDate": "2023-12-20T13:00:00.000Z",`
	`"allowedUntilDate": "2023-12-25T11:00:00.000Z",`
	`"allowedWeekDays": 127,`
	`"smartlockIds": [`
	`"18891899123",`
	`"18891899124"`
	`],`
	`"remoteAllowed": false,`
	`"type": 0`
	`}'`

Key rules of creating auths:

- The time permissions can be provided as per the booking date so that the guests will not be able to access the property outside the permissible time window.
    - The time should be provided in the UTC format
    - Auths are created in the same timezone as that of the Smart Lock, hence the devices should be set to the local timezone
    - allowedFromDate should match the check-in date & time
    - allowedUntilDate should match the check-out date & time
    - Setting allowedUntilDate only disables the auth permission after expiry but doesn't delete it
- Locking/unlocking the devices remotely is not recommended for guests, hence you can set remoteAllowed = false.
- If there are 2 guests, then it is recommended to create 2 auths as guests will receive two separate invitation codes which can be redeemed in the Nuki App (within 48h).

- It is recommended to include the Openers too in the smartlockIds, so that they can be opened from the Nuki App.

8.3 Create a keypad code that can be used by all guests

API endpoint	PUT /smartlock/auth
	{
	"name": "BOOKING123",
	"type": "13",
	"code": "252525",
	"smartlockIds": [
	"18891899123",
	"18891899124"
	],
	"allowedFromDate": "2023-12-20T13:00:00.000Z",
	"allowedUntilDate": "2023-12-25T11:00:00.000Z",
	"allowedWeekDays": 127
	}

curl command	`curl -X 'PUT' \`
	`'https://api.nuki.io/smartlock/auth' \`
	`-H 'accept: application/json' \`
	`-H 'authorization: Bearer ACCESS_TOKEN' \`
	`-H 'Content-Type: application/json' \`
	`-d '{`
	`"name": "BOOKING123",`
	`"allowedFromDate": "2023-12-20T13:00:00.000Z",`
	`"allowedUntilDate": "2023-12-25T11:00:00.000Z",`
	`"allowedWeekDays": 127,`
	`"smartlockIds": [`
	`"18891899123",`
	`"18891899124"`
	`],`
	`"type": 13,`
	`"code": 252525`
	`}'`

Key rules of creating keypad codes:

- The keypad code is a 6-digit code.
- It cannot contain “0” and must not start with “12”.
- The keypad code is also unique per device.
- If you have two devices, one Smart Lock and one Opener, both connected to different keypads, it is recommended to create the same keypad code for both devices (as shown above).

8.4 Delete the auths with the corresponding auth “id” after the booking expires

API endpoint	DELETE /smartlock/auth ["647cfac64dd15972f2122e26", "647cfac64dd15972f2122e34"]

curl command	`curl -X 'DELETE' \`
	`  'https://api.nuki.io/smartlock/auth' \`
	`  -H 'accept: application/json' \`
	`  -H 'authorization: Bearer ACCESS_TOKEN' \`
	`  -H 'Content-Type: application/json' \`
	`  -d '[`
	`"647cfac64dd15972f2122e26",`
	`"647cfac64dd15972f2122e34"`
	`]'`

The clean up of auths is a recommended practice as a maximum of 200 keypad codes can be created per Smart Lock version 3 onwards, and 100 codes per Smart Lock 1/2.

9. Create a keypad code with recurring access time

Let’s consider the use case where the housekeeping staff needs recurring access to the vacation home, from Monday to Saturday, between 10AM to 2PM. It is not feasible to set up new keypad codes for returning users, hence the access times should be set appropriately.

Create a keypad code with unlimited, recurring access time

API endpoint	PUT /smartlock/auth
	{
	"name": "housekeeping",
	"type": "13",
	"code": "292929",
	"smartlockIds": [
	"18891899123",
	"18891899124"
	],
	"allowedFromDate": "2023-12-20T13:00:00.000Z",
	"allowedWeekDays": 126,
	"allowedFromTime": 600,
	"allowedUntilTime": 840
	}

curl command	`curl -X 'PUT' \`

```
  'https://api.nuki.io/smartlock/auth' \
  -H 'accept: application/json' \
  -H 'authorization: Bearer ACCESS_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "housekeeping",
  "allowedFromDate": "2023-12-20T13:00:00.000Z",
  "allowedWeekDays": 126,
  "allowedFromTime": 600,
  "allowedUntilTime": 840,
  "smartlockIds": [
      "18891899123",
      "18891899124"
  ],
  "type": 13,
  "code": 292929
}'
```

How to set the access time:

- allowedWeekDays is set via bitmask

Day	Bitmask	Value
Monday	01000000	64
Tuesday	00100000	32
Wednesday	00010000	16
Thursday	00001000	8
Friday	00000100	4
Saturday	00000010	2
Sunday	00000001	1

    - Monday - Saturday ➜ 64+32+16+8+4+2 = 126
    - This is a mandatory parameter for setting time-restricted permissions
- allowedFromTime is set in minutes from midnight
    - 10AM ➜ 10 hours from midnight i.e. 10*60 = 600
- allowedUntilTime is set in minutes from midnight
    - 2PM ➜ 14 hours from midnight i.e. 14*60 = 840

10. Invite a user without triggering Nuki invite email (via webhooks)

This feature is apt for integrators who want to send the invite email through their PMS system, and do not want to trigger the default Nuki invite email. The Nuki Web Advanced API can be used to receive Nuki invite codes via the webhook and then be used within the PMS channel to send out a customised invite email to their guests. The invite codes still need to be redeemed from the Nuki App.

To invite a user through the advanced API, call the advanced auth creation endpoint

API endpoint

PUT /smartlock/auth/advanced

```
{
  "name": "BOOKING123 #2",
   "accountUserId": "1264915785",
   "type": "0",
   "smartlockIds": [
     "18891899123"
      ],
   "remoteAllowed": false,
   "allowedFromDate": "2023-12-20T13:00:00.000Z",
   "allowedUntilDate": "2023-12-25T11:00:00.000Z",
   "allowedWeekDays": 127
}
```

curl command

```
curl -X 'PUT' \
  'https://api.nuki.io/smartlock/auth/advanced' \
  -H 'accept: application/json' \
  -H 'authorization: Bearer ACCESS_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "BOOKING123 #2",
  "accountUserId": 1264915785,
  "allowedFromDate": "2023-12-20T13:00:00.000Z",
  "allowedUntilDate": "2023-12-25T11:00:00.000Z",
  "allowedWeekDays": 127,
  "smartlockIds": [
      "18891899123"
  ],
  "remoteAllowed": false,
  "type": 0
}'
```

With the Advanced API access, you can receive an API-triggered webhook with the invite code for the auth created

Webhook

```
{
```

response


```
"headers": {},
 "body": {
   "inviteCode": "ALE-OGB-VVA",
   "detail": [
     {
       "smartlockId": 18891899123,
       "success": true,
       "id": "6597f599512b3f6c71b5b4ab",
       "authId": 3431775
     }
   ],
   "type": "AuthCreation",
   "requestId": "6597f599512b3f6c71b5b4a8",
   "success": true
 },
 "timestamp": "2024-01-05T12:27:06.116Z",
 "path": "WEBHOOK_URL"
}
```


When the guest redeems the inviteCode "ALE-OGB-VVA" in the Nuki App, they will have permissions to lock/unlock the device during the configured access time.

BEST PRACTICES

I. How can I check if an auth has been created successfully?

If you are not using webhooks:
The Nuki Web API is asynchronous. When you send a command, you will get an immediate OK response (i.e. 200) which means that the request has succeeded, but that doesn't guarantee that the client has executed the request successfully. Once the server has received the API call, it will try to reach the Smart Lock and create the code (all the codes are stored on the Smart Lock itself). This usually takes some seconds, and the device should be online. Thus, call the GET /smartlock/auth endpoint within a few minutes to validate if the auth was created successfully.

If you are using webhooks:
A webhook is triggered when an authorization is created (also when it is modified or deleted). Example webhook response:


```
AuthResponse {
"feature": "DEVICE_AUTHS",
    "deleted": false,
    "smartlockAuth": {
      "id": "65219648fe57075d71a6cb01",
      "smartlockId": 11000009999,
      "authId": 1,
```

```
    "code": 982345,
    "type": 13,
    "name": "example",
     … }
}
```

Thus, the best practice is to use webhooks to validate if auths are created successfully.

II. How often are the auths synced to the device?

When the auth is created through the API, it syncs it on the device immediately if the device is online. If this fails for some reason, then there is a daily sync (runs once every 24 hours) which syncs all the auths once per day (device must be online).

You can also perform a forced sync via the API by calling POST /smartlock/{smartlockId}/sync but manual syncing should be avoided as it will drain the devices batteries and can lead to errors when overloading the device or Nuki servers.

NOTE: Authorization changes or master data changes done via Nuki App or the BLE API, as well as authorizations created via direct pairing will only trigger webhooks with the next Nuki Web sync.

III. When should I sync a device and when I should not?

When to sync?

- The serverState seems to be not up-to-date.
- An unknown authorization ID in a log entry which needs to be mapped immediately to not break a user interface.

When not to sync?

- In regular intervals “as a general fallback”: Webhooks should be used to avoid polling the Web API and not as an additional feature (to even increase the load on the servers).
- If automated 24-hour sync fails regularly (i.e. no "DEVICE_STATUS" webhook for >24 hours.): This is most likely an issue with the Bridge-connection which should be addressed and not covered by more syncing.

IV. How should I set up my webhooks?

Monitor the webhook URL and map payload content according to your needs. Deactivate unnecessary features to reduce the number of webhooks sent to a minimum.

If you want to explore all possible options you can start with all features activating and clustering information received by the different types, as all webhook payloads contain the feature which triggered them, e.g. feature = "DEVICE_LOGS". From there you can deactivate

unneeded features to reduce the number of webhook notifications and duplicates for your integration.

Note that the changed values are not additionally highlighted in the payload and have to be monitored on your side.

V. How should I call the API endpoints?

We've implemented the Cross-Origin Resource Sharing (CORS) policy which allows access only from a strict list of trusted domains and subdomains to match against, and doesn't support the "null" origin.

You can proxy the request through your own server so that the server can make the request to 'https://api.nuki.io' on behalf of your web application.

CORE CONCEPTS

A. Convert Device ID from HEX to DECIMAL

How to get the HEX Device ID in decimal format?

Device IDs are normally represented in the HEX (hexadecimal) format in the Nuki App or Web and on the device itself. The Nuki Web API expects the Device IDs to be sent as an integer (or decimal format). You can use a regular calculator (example) to convert the HEX ID to Decimal ID, but before doing so, the type of the device has to be prefixed to the HEX ID.

For example, for a Smart Lock 2, prefix type '0' to the Hex ID and convert it to decimal, as shown below.

Device	Type	displayed Hex ID	calculated Hex ID (with prefix)	calculated Decimal ID
Nuki Smartlock 1 or 2	0	1A2B3C4D	1A2B3C4D	439041101
Nuki Box	1	1A2B3C4D	11A2B3C4D	4734008397
Nuki Opener	2	1A2B3C4D	21A2B3C4D	9028975693
Nuki Smartdoor	3	1A2B3C4D	31A2B3C4D	13323942989
Nuki Smartlock 3 (Basic & Pro)	4	1A2B3C4D	41A2B3C4D	17618910285
Nuki Smartlock 4th Gen (Basic & Pro)	4	1A2B3C4D	41A2B3C4D	17618910285

Nuki Smartlock Ultra	5	1A2B3C4D	51A2B3C4D	21913877581

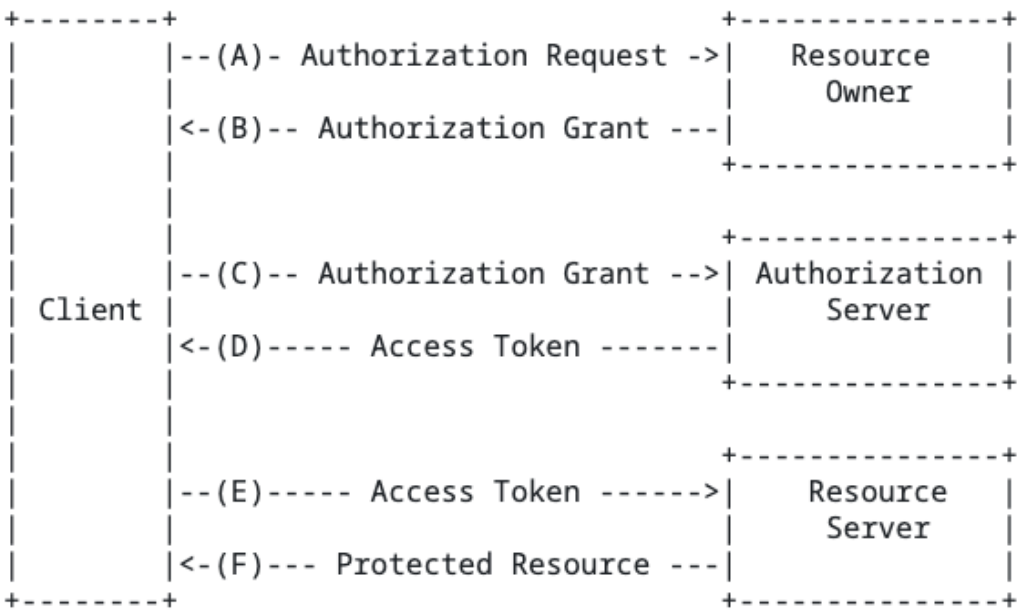
B. OAuth 2.0

OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. While we've provided the concept at a very high level below, you can refer to more information on the OAuth2.0 website.

OAuth 2 defines four roles:

#	Role	Description	In the context of Nuki
1	Resource owner	This is the user who is capable of granting access to an application	Customer with a Nuki Web account
2	Client	This is the application that wants to access the user's account	Client application of the partner or integrator, for e.g. Magenta
3	Resource server	This is the server that hosts the protected user accounts	Nuki authorization server
4	Authorization server	This is the server that verifies the identity of the user and then issues access tokens to the application	Partner server

Below is the abstract protocol flow to obtain an **access token** (a unique security credential that identifies the user):



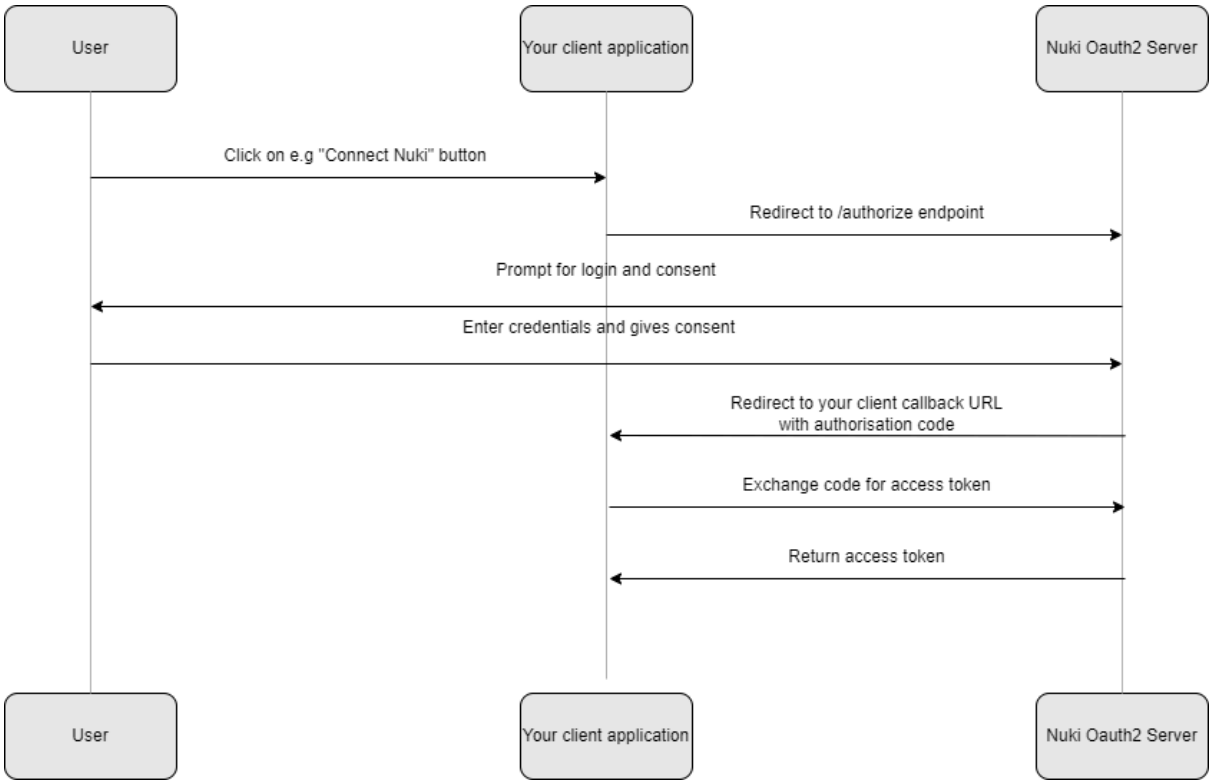
There are multiple authorization grant types within OAuth2.0 but Nuki supports the authorization code flow that is used with server side applications. Refer to Section 5.1 Authorization Code Flow for more information on Authorization Code Flow.

C. Authorization Code Flow

The Authorization Code Flow is the most secure and widely used OAuth 2 flow for web applications.

At a high level:

- The user authorizes a client application (for example, Magenta, or Beds24) to provide access to their Nuki Web account by clicking on a link or button within the application.
- The user is redirected to the Nuki authorization server.
- When the user authenticates themselves successfully, they grant permission (through scopes) to your application.
- The Nuki authorization server generates an authorization code and redirects the user back to your application with the authorization code.
- Your application exchanges the authorization code for an access token that can be used to access the resource (in this case, user's Nuki Web account).
- Since the access token is only valid for a short period of time, a refresh token is provided that can be used to renew the access token.



D. Bearer Token

Bearer authentication (also called token authentication) is an HTTP authentication scheme that involves security tokens called bearer tokens. The name “Bearer authentication” can be understood as “give access to the bearer of this token.” The bearer token is a cryptic string, usually generated by the server in response to a login request. The client must send this token in the Authorization header when making requests to protected resources: Authorization: Bearer <ACCESS_TOKEN>. You can find more information on the Swagger website.

Example curl call:

```
curl -X GET --header 'Accept: application/json' --header
'Authorization: Bearer ACCESS_TOKEN'
'https://api.nuki.io/smartlock'
```

E. Central & Decentral Webhooks

Nuki makes it possible to integrate the Nuki infrastructure to any third party integrator. This empowers any company to implement their business needs in companionship with Nuki devices. To asynchronously inform about any device change in a timely manner Nuki supports webhooks.

There are two different kind of webhooks regarding the triggering entity:

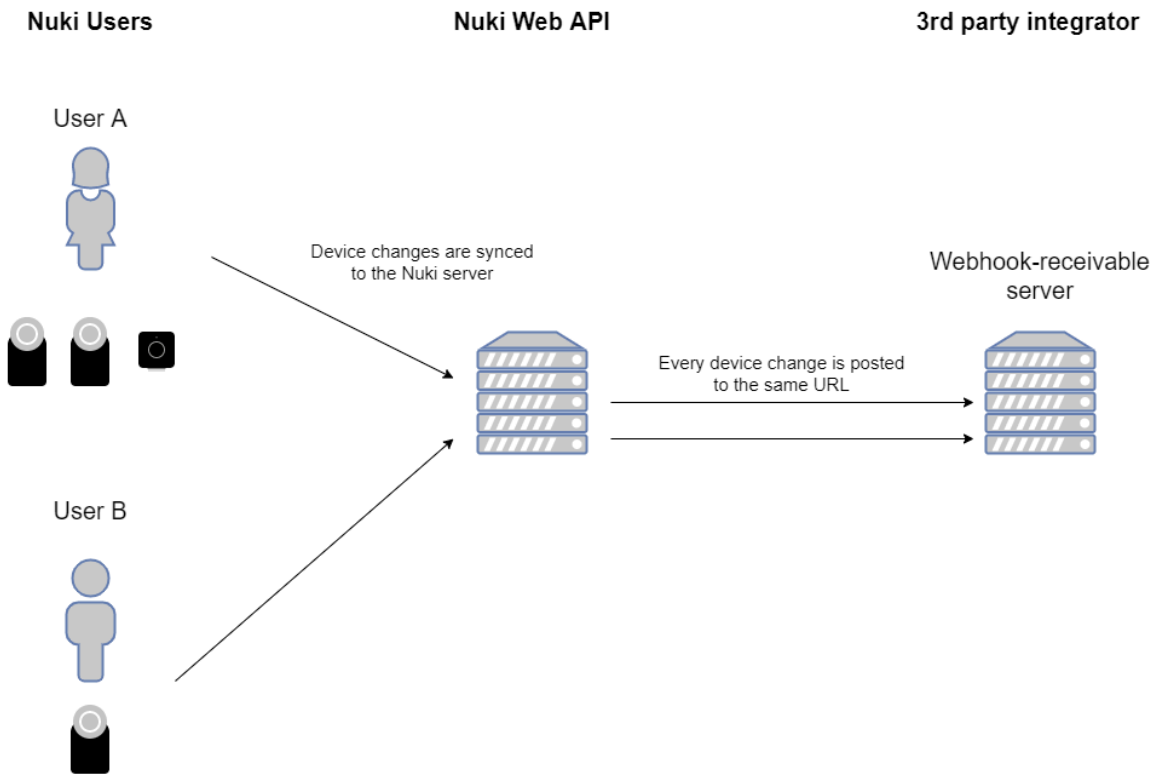
- **Device-triggered webhooks**: A webhook created by the Nuki device itself, for e.g. lock action from Nuki App, updated config due to a sync, etc.
- **API-triggered webhooks**: A webhook as response from an endpoint in the Nuki Advanced API (https://api.nuki.io/#/AdvancedApi)

An integrator can choose between two workflows:

- **Central webhooks**: Nuki posts webhooks to one distinct URL
- **Decentral webhooks**: Nuki posts webhooks to several different URLs

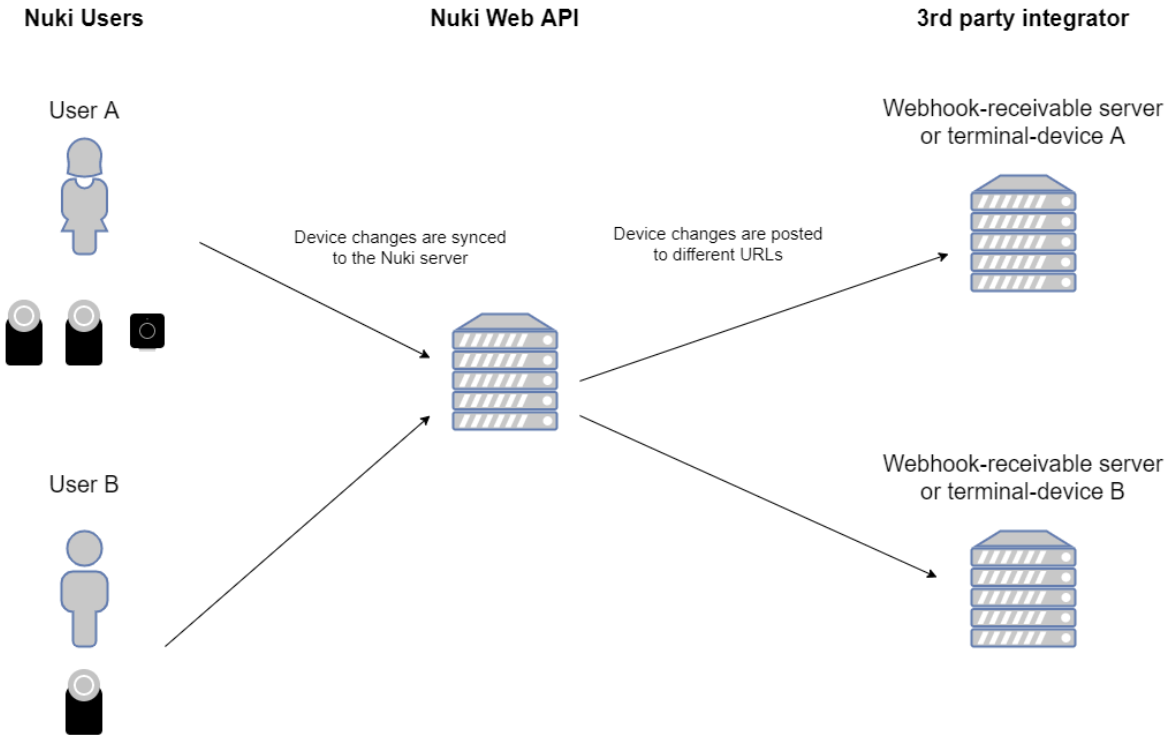
Central webhook workflow

This workflow forwards webhooks to a single URL endpoint. This workflow is best suited for environments with one receiving instance, which handles webhooks for all Nuki users.



Decentral webhook workflow

This workflow is for integrators which want to obtain webhooks on different URLs. This use case is best suited for an integrator which has terminal-devices for each Nuki user separately.



F. How to monitor Webhooks

A webhook is always assigned to exactly one API key and stored as a log entry. The logs can be viewed in Nuki Web or accessed via the endpoint https://api.nuki.io/api/key/{apiKeyId}/webhook/logs.

To retrieve the logs for a webhook, call the GET endpoint below:

```
GET https://api.nuki.io/api/key/{apiKeyId}/webhook/logs
?id=[Optionally filter for older logs]
&limit=[Amount of logs (default: 50, max: 100)]
```

Header:

```
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Authorization: Basic [Base64enc("ClientID":"ClientSecret")]
```

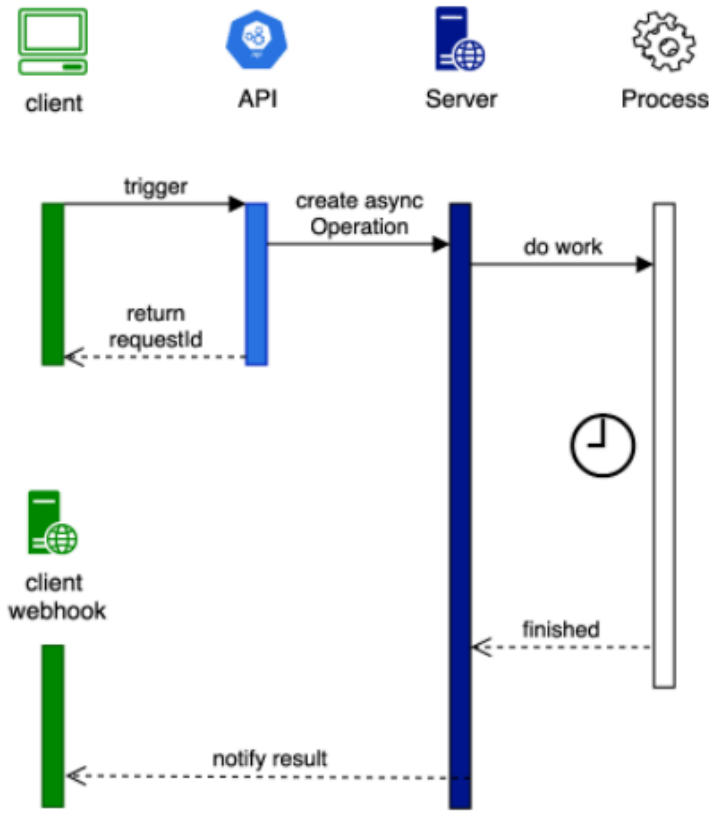
- Only the last 300 messages are stored.
- The logs are returned in chronologically descending order.
- If no parameters are specified, the last max. 50 logs are returned.
- By specifying the "Id" parameter all logs that were after this specific Id are returned.
    - The Id is a hexadecimal representation of an Object ID.
- With the limit parameter you can set the number of logs returned from 1-100 (any other value is ignored and the default value of 50 is assumed).

In case you change the webhook URL or experience issues on your side, you can also directly monitor sent webhooks via the logs at https://web.nuki.io/#/pages/web-api to be able to quickly track down issues with undelivered webhooks.

Changing the webhook URL can also be done directly via the Web API with POST /api/key/{apiKeyId}/advanced.

G. API-triggered Webhook Responses

When an Advanced API endpoint is called, a response is returned synchronously with a request ID. At the same time, the request is processed asynchronously on the server and a result is returned to a user defined webhook after the asynchronous process is completed.



There are two types of result responses:

- Response for a single device
- Response for multiple devices

Single result response:

For ‘LockAction', 'UnlockAction' and 'SmartlockAction’:

```
WebhookResponse {
type (WebhookType): type = ['LockAction', 'UnlockAction',
'SmartlockAction'],
requestId (string): Returned RequestId after Api was triggered,
smartlockId (long): The Id of the affected device,
success (boolean): True if the action was performed successfully,
errorCode (String): [Optional] Indicates the error (hexadecimal
representation) in case of failure
}
```

Multiple result response:

For 'AuthCreation':

```
WebhookResponse {
inviteCode (string): invite Code
detail: (DetailItem[])[
smartlockId (long): The Id of the affected device,
```

```
success (boolean): True if the creation was performed
successfully,
errorCode (Integer): [Optional] Indicates the error in case of
failure
],
type (WebhookType): type = ['AuthCreation'],
requestId (string): Returned RequestId after Api was triggered,
success (boolean): True if the action was performed successfully,
errorCode (Integer): [Optional] Indicates the error in case of
failure
}
```

H. Device-triggered Webhook Responses

The device-triggered webhooks work like push notifications. In total, there are 6 features and the trigger events are described below:

Webhook feature	Triggered when...
Device status	the status of the device has changed
Device master data	the master data of the device has changed or a sync has been triggered
Device configs	the config of the device has changed or a sync has been triggered
Device logs	an activity log entry is created
Device authorizations	an authorization is created, modified or deleted
Account user	a Nuki Web user is created, modified or deleted

1. Device status

```
StateResponse {
      feature = "DEVICE_STATUS",
      smartlockId (long): The Id of the affected device,
      state (Smartlock.state): {
      mode (integer): The smartlock mode: 0 .. uninitialized, 1 ..
      pairing, 2 .. door (default), 3 .. continuous (type=2 only),
      4 .. maintenance,
      state (integer): The smartlock state: type=0/3/4: 0 ..
      uncalibrated, 1 .. locked, 2 .. unlocking, 3 .. unlocked, 4
      .. locking, 5 .. unlatched, 6 .. unlocked (lock 'n' go), 7 ..
      unlatching, 224 .. Error wrong entry code, 225 .. Error wrong
      Fingerprint, 254 .. motor blocked, 255 .. undefined; type=2:
```

0 .. untrained, 1 .. online, 3 .. ring to open active, 5 .. open, 7 .. opening, 253 .. boot run, 255 .. undefined,
trigger (integer): The state trigger: 0 .. system, 1 .. manual, 2 .. button, 3 .. automatic, 4 .. web (type=1 only), 5 .. app (type=1 only), 6 .. continuous mode (type=2 only), 7 .. accessory (type=3 only),
lastAction (integer): The action: type=0/3/4: 1 .. unlock, 2 .. lock, 3 .. unlatch, 4 .. lock 'n' go, 5 .. lock 'n' go with unlatch; type=1: 1 .. unlock; type=2: 1 .. activate ring to open, 2 .. deactivate ring to open, 3 .. open (electric strike actuation),
batteryCritical (boolean): True if the battery state of the device is critical,
batteryCharging (boolean): True if a Nuki battery pack in a Smart Lock is currently charging,
batteryCharge (integer): Remaining capacity of a Nuki battery pack in %,
keypadBatteryCritical (boolean): True if the battery of a paired Keypad is critical,
doorsensorBatteryCritical (boolean): True if the battery of a paired door sensor is critical,
doorState (integer): The door state: 0 .. unavailable/not paired, 1 .. deactivated, 2 .. door closed, 3 .. door opened, 4 .. door state unknown, 5 .. calibrating, 16 .. uncalibrated, 240 .. removed, 255 .. unknown,
ringToOpenTimer (integer): Remaining ring to open time; 0 if ring to open is not active (type=2 only),
nightMode (boolean): True if night mode currently active},
serverState (integer): The server state: 0 .. ok, 1 .. unregistered, 2 .. auth uuid invalid, 3 .. auth invalid, 4 .. offline,
adminPinState (integer): The admin pin state: 0 .. ok, 1 .. missing, 2 .. invalid}

2. **Device master data**

MasterDataResponse {
feature = "DEVICE_MASTERDATA",
deleted (boolean): flag if the device is deleted or not ,
smartlockId (long): The Id of the affected device ,
accountId (integer): The Id of the affected account ,
type (integer): Type of the device ,
authId (integer): The authorization id ,
name (string): The name of the smartlock ,
favorite (boolean): The favorite flag ,
firmwareVersion (integer, optional, read only): The firmware version ,

```
      hardwareVersion (integer, optional, read only): The hardware
      version ,
      serverState (integer): The server state: 0 .. ok, 1 ..
      unregistered, 2 .. auth uuid invalid, 3 .. auth invalid, 4 ..
      offline ,
      adminPinState (integer): The admin pin state: 0 .. ok, 1 ..
      missing, 2 .. invalid ,
      creationDate (string, optional): The creation date ,
      updateDate (string, optional): The update date
}
```

Firmware update

Notifications for changes to the firmware version of a device can be retrieved by setting the feature "DEVICE_MASTERDATA". The value for firmware Version is provided as an integer which has to be transformed into HEX format to show the current version.

For example:
firmwareVersion = 133135
133135 (DEC) = 2080F (HEX) = FW v. 2.8.15

Admin PIN errors

Changing the Admin PIN on a device also restricts all actions from the Nuki Web (API) for which administration rights are needed. Therefore it is recommended to only change it within the Nuki Web. Changes done outside of Nuki Web can be retrieved by setting the feature "DEVICE_MASTERDATA".

The adminPinState can be used to track if functionality may be missing due to a change here. Possible values of adminPinState:

- 0 .. ok
- 1 .. missing - no Admin PIN set; recommended to add one for security reasons
- 2 .. invalid - new Admin PIN needs to be set in Nuki Web

3. Device configs

```
ConfigResponse {
      feature = "DEVICE_CONFIG",
      smartlockId (long): The Id of the affected device,

      config (Smartlock.config): {
      name (string): The name of the smartlock for new users,
      latitude (float): The latitude of the smartlock position,
      longitude (float): The longitude of the smartlock position,
      autoUnlatch (boolean): True if the door should be unlatched
      on unlocking (knob) (only for type=1 and type=3),
      liftUpHandle (boolean): True if the door has a lift up
      handle, which is required to be lifted up to lock the door,
```

pairingEnabled (boolean): True if the pairing is allowed via the smartlock button,
buttonEnabled (boolean): True if the button on the smartlock is enabled,
ledEnabled (boolean): True if the LED on the smartlock is enabled,
ledBrightness (integer): The brightness of the LED: 0 .. off, 5 .. max (only for type=1 and type=3),
timezoneOffset (integer): The timezone offset (in minutes),
daylightSavingMode (integer): The daylight saving mode: 0 .. off, 1 .. european,
fobPaired (boolean): True if a fob is paired with the smartlock,
fobAction1 (integer): The fob action if button is pressed once: type=0/3/4: 0 .. none, 1 .. unlock, 2 .. lock, 3 .. lock 'n' go, 4 .. intelligent (lock/unlocked based on the current state); type=2: 0 .. none, 1 .. toggle ring to open, 2 .. activate ring to open, 3 .. deactivate ring to open, 7 .. open (electric strike actuation), 8 .. ring,
fobAction2 (integer): The fob action if button is pressed twice,
fobAction3 (integer): The fob action if button is pressed 3 times,
singleLock (boolean): True if the smartlock should only lock once (instead of twice) (only for type=1),
advertisingMode (integer): The advertising mode (battery saving): 0 .. automatic, 1 .. normal, 2 .. slow, 3 .. slowest,
keypadPaired (boolean): True if a keypad is paired with the smartlock,
keypad2Paired (boolean): True if a keypad 2 is paired with the smartlock,
homekitState (integer): The homekit state: 0 .. unavailable, 1 .. disabled, 2 .. enabled, 3 .. enabled & paired,
matterState (integer): The matter state: 0 .. not available, 1 .. disabled and no certificate available, 2 .. disabled, 3 .. enabled, 4 .. enabled & paired ,
timezoneId (integer): The timezone id ,
deviceType (integer): The device type of a Nuki device ,
wifiEnabled (boolean): Flag that indicates if the device's internal WIFI module can be used},
productVariant (integer) : The product variant for Smartlock 5: 1 .. Go, 2 .. Pro, 3 .. Ultra

advancedConfig (Smartlock.AdvancedConfig): {
totalDegrees (integer): The absolute total position in degrees that has been reached during calibration,

singleLockedPositionOffsetDegrees (integer): Offset that alters the single locked position,
unlockedToLockedTransitionOffsetDegrees (integer): Offset that alters the position where transition from unlocked to locked happens,
unlockedPositionOffsetDegrees (integer): Offset that alters the unlocked position,
lockedPositionOffsetDegrees (integer): Offset that alters the locked position,
detachedCylinder (boolean): Flag that indicates that the inner side of the used cylinder is detached from the outer side,
batteryType (integer): The type of the batteries present in the smart lock: 0 .. alkali, 1 .. accumulator, 2 .. lithium,
autoLock (boolean): New separate flag with FW >= 2.7.8/1.9.1: The Auto Lock feature automatically locks your door when it has been unlocked for a certain period of time,
autoLockTimeout (integer): Seconds until the smart lock relocks itself after it has been unlocked. FW < 2.7.8/1.9.1: No auto relock if value is 0, FW >= 2.7.8/1.9.1: has to be >=2 (defaults to 2 for values <2 if autoLock is set to true),
autoUpdateEnabled (boolean): Flag that indicates if available firmware updates for the device should be installed automatically,
lngTimeout (integer): Timeout in seconds for lock ‘n’ go [5, 10, 15, 20, 30, 45, 60],
singleButtonPressAction (integer): The desired action, if the button is pressed once: 0 .. no action, 1 .. intelligent, 2 .. unlock, 3 .. lock, 4 .. unlatch, 5 .. lock 'n' go, 6 .. show status
doubleButtonPressAction (integer): The desired action, if the button is pressed twice: 0 .. no action, 1 .. intelligent, 2 .. unlock, 3 .. lock, 4 .. unlatch, 5 .. lock 'n' go, 6 .. show status,
automaticBatteryTypeDetection (boolean): Flag that indicates if the automatic detection of the battery type is enabled,
unlatchDuration (integer): Duration in seconds for holding the latch in unlatched position [1, 3, 5, 7, 10, 15, 20, 30]},

openerAdvancedConfig (Smartlock.OpenerAdvancedConfig): {
intercomId (integer): The database ID of the connected intercom
busModeSwitch (integer): Method to switch between data and analogue mode [0,1],
shortCircuitDuration (integer): Duration of the short circuit for BUS mode switching in ms,

electricStrikeDelay (integer): Delay of electric strike activation in ms after lock action 3 'electric strike actuation',
randomElectricStrikeDelay (boolean): Random electricStrikeDelay (range 3000 - 7000 ms) in order to simulate a person inside actuating the electric strike,
electricStrikeDuration (integer): Duration in ms of electric strike actuation lock action 3 'electric strike actuation',
disableRtoAfterRing (boolean): Flag to disable RTO after ring,
rtoTimeout (integer): After this period of time in minutes, RTO gets deactivated automatically,
doorbellSuppression (integer): The doorbell suppression bitmask: first bit (least significant) .. whenever the doorbell rings and CM and RTO are inactive, second bit .. RTO is active, third bit .. CM is active,
doorbellSuppressionDuration (integer): Duration in ms of doorbell suppression (only in Operating mode 2 'digital Intercom'),
soundRing (integer): The sound for ring: 0 .. no sound, 1 .. Sound1, 2 .. Sound2, 3 .. Sound3,
soundOpen (integer): The sound for open: 0 .. no sound, 1 .. Sound1, 2 .. Sound2, 3 .. Sound3,
soundRto (integer): The sound for RTO: 0 .. no sound, 1 .. Sound1, 2 .. Sound2, 3 .. Sound3,
soundCm (integer): The sound for CM: 0 .. no sound, 1 .. Sound1, 2 .. Sound2, 3 .. Sound3,
soundConfirmation (integer): The sound confirmation: 0 .. no sound, 1 .. sound,
soundLevel (integer): The sound level,
singleButtonPressAction (integer): The desired action, if the button is pressed once: 0 .. no action, 1 .. toggle RTO, 2 .. activate RTO, 3 .. deactivate RTO, 4 .. toggle CM, 5 .. activate CM, 6 .. deactivate CM, 7 .. open,
doubleButtonPressAction (integer): The desired action, if the button is pressed twice: 0 .. no action, 1 .. toggle RTO, 2 .. activate RTO, 3 .. deactivate RTO, 4 .. toggle CM, 5 .. activate CM, 6 .. deactivate CM, 7 .. open,
batteryType (integer): The type of the batteries present in the smart lock: 0 .. alkali, 1 .. accumulator, 2 .. lithium, 3 .. fixed,

automaticBatteryTypeDetection (boolean): Flag that indicates if the automatic detection of the battery type is enabled
autoUpdateEnabled (boolean): Flag that indicates if available firmware updates for the deviceshould be installed automatically},

smartdoorAdvancedConfig (Smartlock.SmartdoorAdvancedConfig): {
lngTimeout (integer): Timeout in seconds for lock ‘n’ go [5, 10, 15, 20, 30, 45, 60],
singleButtonPressAction (integer): The desired action, if the button is pressed once: 0 .. no action, 1 .. intelligent, 2 .. unlock, 3 .. lock, 4 .. unlatch, 5 .. lock 'n' go, 6 .. show status,
doubleButtonPressAction (integer): The desired action, if the button is pressed twice: 0 .. no action, 1 .. intelligent, 2 .. unlock, 3 .. lock, 4 .. unlatch, 5 .. lock 'n' go, 6 .. show status,
automaticBatteryTypeDetection (boolean): Flag that indicates if the automatic detection of the battery type is enabled,
unlatchDuration (integer): Duration in seconds for holding the latch in unlatched position [1, 3, 5, 7, 10, 15, 20, 30],
buzzerVolume (integer): The volume of the buzzer: 0 .. off, 1 .. low, 2 .. normal,
supportedBatteryTypes (integer): Set of supported battery types: 0 .. alkali, 1 .. accumulator, 2 .. lithium, 3 .. fixed, 254 .. automatic, 255 .. unknown,
batteryType (integer): The type of the batteries present in the smart lock: 0 .. alkali, 1 .. accumulator, 2 .. lithium, 3 .. fixed, 255 .. unknown,
autoLockTimeout (integer): Seconds until the smart lock relocks itself after it has been unlocked. No auto relock if value is 0,
autoLock (boolean): The Auto Lock feature automatically locks your door when it has been unlocked for a certain period of time}
}

4. **Device logs**

LogResponse {
feature = "DEVICE_LOGS",
smartlockId (long): The Id of the affected device,
deviceType (integer): The device type: 0 .. smartlock and box, 2 .. opener, 3 .. smartdoor,
name (string): The name,
action (integer): The action: 1 .. unlock, 2 .. lock, 3 .. unlatch, 4 .. lock'n'go, 5 .. lock'n'go with unlatch, 208 .. door warning ajar, 209 door warning status mismatch, 224 .. doorbell recognition (only Opener), 240 .. door opened, 241 .. door closed, 242 .. door sensor jammed, 243 .. firmware update, 250 .. door log enabled, 251 .. door log disabled,

```
      252 .. initialization, 253 .. calibration, 254 .. log
      enabled, 255 .. log disabled,
      trigger (integer): The trigger: 0 .. system, 1 .. manual, 2
      .. button, 3 .. automatic, 4 .. web, 5 .. app, 6 .. auto
      lock, 7 .. accessory, 255 .. keypad,
      state (integer): The completion state: 0 .. Success, 1 ..
      Motor blocked, 2 .. Canceled, 3 .. Too recent, 4 .. Busy, 5
      .. Low motor voltage, 6 .. Clutch failure, 7 .. Motor power
      failure, 8 .. Incomplete, 9 .. Rejected, 10 .. Rejected night
      mode, 254 .. Other error, 255 .. Unknown error,
      autoUnlock (boolean): True if it was an auto unlock,
      date (string): The log date,

      openerLog  SmartlockLog.OpenerLog: {
      activeCm (boolean): Flag indicating if continuous mode was
      active,
      activeRto (boolean): Flag indicating if ring to open was
      active,
      source (integer): The cause of the activation of ring to open
      or continuous mode: 0 .. doorbell, 1 .. timecontrol, 2 ..
      app, 3 .. button, 4 .. fob, 5 .. bridge, 6 .. keypad,
      flagGeoFence (boolean): Flag indicating a geo fence induced
      action,
      flagForce (boolean): Flag indicating a force induced action,
      flagDoorbellSuppression (boolean): Flag indicating if doorbell
      suppression was active}
}
```

5. **Device authorizations**

```
AuthResponse {
      feature = "DEVICE_AUTHS",
      deleted (boolean): flag if the auth is deleted or not,
      smartlockAuth (SmartlockAuth): {
      id (string):The unique id for the smartlock authorization,
      smartlockId (integer): The smartlock id,
      accountUserId (integer): The id of the linked account user,
      authId (integer): The smartlock authorization id,
      code (integer): The keypad code (only for type keypad),
      type (integer): The type of the authorization: 0 .. app, 1 ..
      bridge, 2 .. fob, 3 .. keypad, 13 .. keypad code, 14 ..
      z-key, 15 .. virtual,
      name (string): The name of the authorization (max 32 chars),
      enabled (boolean): True if the auth is enabled,
      remoteAllowed (boolean): True if the auth has remote access,
      lockCount (integer): The lock count,
      allowedFromDate (string): The allowed from date,
```

```
      allowedUntilDate (string): The allowed until date,
      allowedWeekDays (integer): The allowed weekdays bitmask: 64
      .. monday, 32 .. tuesday, 16 .. wednesday, 8 .. thursday, 4
      .. friday, 2 .. saturday, 1 .. sunday, (mandatory for setting
      time-limited access)
      allowedFromTime (integer): The allowed from time (in minutes
      from midnight),
      allowedUntilTime (integer): The allowed until time (in
      minutes from midnight),
      creationDate (string): The creation date,
      updateDate (string): The update date}
}
```

6. **Account user**

```
AccountUserResponse {
      feature = "ACCOUNT_USER",
      deleted (boolean): flag if the account user is deleted or
      not,
      accountUserId (integer): The account user id,
      accountId (integer): The account id,
      type (integer): The optional type: 0 .. user, 1 .. company,
      email (string): The email address,
      name (string): The name,
      language (string): The language code,
      creationDate (string): The creation date,
      updateDate (string): The update date
}
```

I. Rate Limits

Nuki Web API is accessible publicly by thousands of developers. In order to limit the volume of requests and prevent system abuse, limits are placed on the number of requests that can be made to the API. Once a rate limit has been reached, further requests are rejected until the limit expires.

Webhook error rate limit:
If the error rate on a webhook exceeds 5% in the last 24 hours, a warning email is sent to the developer's email address that is registered with the Nuki Web Advanced API. Therefore it is important to set an email which is reachable and will be checked regularly.

A failed attempt is only registered as failed if the HTTP status code of the webhook is not 200 OK, 202 Accepted or 204 No-Content.

Nuki may suspend the webhooks service for an URL if the error rate is 100% for an extended period of time.

DEFINITIONS

J. Scopes

A scope is a permission that is set on a token, a context in which that token may act. When certain scopes are not set on a token, that token is not permitted to perform those operations.

Scope	Scope Name	Scope Description
account	View and manage account	• Edit the Nuki Web user • Create, edit and delete Nuki Web sub-users • Create, edit and delete API keys
notification	Notification	• Enable, disable, edit notifications for Nuki devices
smartlock	View and edit devices	• Add, view, edit and remove devices to/from Nuki Web (API)
smartlock.readonly	View devices	• Show Nuki devices in Nuki Web (API)
smartlock.action	Operate devices	• Operate devices via Nuki Web (API)
smartlock.auth	View and manage authorisations	• Create, edit and delete authorizations on a Nuki device via Nuki Web (API)
smartlock.config	Manage device configuration	• Change device settings in Nuki Web (API)
smartlock.log	View activity logs and get log notifications	• Retrieve logs from Nuki devices via Nuki Web (API) • Manage webhooks for the Web API

For example, a token with the **scope:smartlock.readonly** is permitted to only view the devices, and the **scope:smartlock** is permitted to view as well as edit the devices.

K. Nuki Web API Endpoints

The Nuki Web API endpoints are documented in Swagger.
For your reference, the endpoints are provided below along with their usage.

Path	Usage	Available endpoints	Description	Scopes needed
Account	Nuki Web account	POST, GET, PUT, DELETE	Handle Nuki Web accounts and sub-accounts, OTP settings and password reset	account
AccountSubscription	Nuki Box subscriptions	POST, GET	Check and edit Nuki Box subscription tokens	account
AccountUser	Nuki device users	POST, GET, PUT, DELETE	Create, edit and delete (email based) Nuki device users to which authorizations can be assigned	account, smartlock.auth
Address	Nuki device grouping	POST, GET	Connecting an array of Nuki devices to an address object for Nuki Box subscriptions and short rental	account
AddressReservation	Short rental integration	GET, POST	Handle bookings for connected listings from short rental integrations	account
AddressToken	Nuki Box subscriptions	POST, GET	Create and check Nuki Box subscriptions	-
ApiKey	Manage Web API keys	POST, GET, PUT, DELETE	Create, edit and delete API keys for the Nuki Web API	account
Company	Nuki Partner network	GET	List companies from Nukis partner network.	-
Opener	Opener compatibility check and installation	GET	List Opener compatible intercoms per brand.	-

Service	Short Rental	POST, GET	Link, unlink and sync available short rental integration services	account
Smartlock	Nuki devices	POST, GET, PUT, DELETE	Manage Nuki devices and device settings	smartlock, (smartlock.read Only)
SmartlockAuth	Authorizations	POST, GET, PUT, DELETE	Create, edit and delete authorizations on Nuki devices	smartlock.auth
SmartlockLog	Activity Log	GET	Retrieve log files from Nuki devices	smartlock.log
Subscription	Nuki Box subscriptions	GET	Check for valid Nuki Box subscriptions	account

L. Smart Lock States

Below are the definitions for the most important states of the devices, also documented here: https://api.nuki.io/static/swagger/swagger.json

Name	Smartlock	Opener	Smart Door
mode	The current operation state of the Nuki Smart Lock 0 uninitialized 1 pairing 2 door (default) 3 - 4 maintenance 5 -	The current operation state of the Nuki Opener 0 uninitialized 1 pairing 2 door (default) 3 continuous 4 maintenance 5 -	The current operation state of the Nuki Smart Door 0 uninitialized 1 pairing 2 door (default) 3 failure 4 maintenance 5 test

state	The current state of the Nuki Smart Lock 0 uncalibrated 1 locked 2 unlocking 3 unlocked 4 locking 5 unlatched 6 unlocked (lock'n'go) 7 unlatching 253 boot run 254 motor blocked 255 undefined	The current state of the intercom control within Nuki Opener 0 untrained 1 online 2 - 3 rto active 4 - 5 open 6 - 7 opening 253 boot run 254 - 255 undefined	The current state of the Nuki Smart Door 0 not activated 1 locked 2 unlocking 3 unlocked 4 locking 5 unlatched 6 unlocked (lock'n'go) 7 unlatching 253 boot run 254 motor blocked 255 undefined
trigger	The trigger, that caused the state change within the Nuki Smart Lock 0 system (bluetooth) 1 manual 2 button 3 automatic 4 - 5 - 6 - 7 - *4 & 5 are Box-only*	The trigger, that caused the state change within the Nuki Opener 0 system (bluetooth) 1 manual 2 button 3 automatic 4 - 5 - 6 continuous mode 7 - *4 & 5 are Box-only*	The trigger, that caused the state change within the Nuki Smart Door 0 system (bluetooth) 1 manual 2 button 3 automatic 4 - 5 - 6 auto lock 7 external accessory *4 & 5 are Box-only*
lastAction	1 unlock 2 lock 3 unlatch 4 lock'n'go 5 lock'n'go with unlatch 6 - 7 -	1 activate rto 2 deactivate rto 3 electric strike actuation 4 - 5 - 6 activate cm 7 deactivate cm	1 unlock 2 lock 3 unlatch 4 lock'n'go 5 lock'n'go with unlatch 6 - 7 -

M. Smart Lock Actions

Lock actions are used as a parameter in lock commands, and are also available as lastAction for states or state changes.

Name	Smartlock, Smart Door	Opener	Box
action	1 unlock 2 lock 3 unlatch 4 lock'n'go 5 lock'n'go with unlatch 6 - 7 -	1 activate rto 2 deactivate rto 3 electric strike actuation 4 - 5 - 6 activate cm 7 deactivate cm	1 unlock 2 - 3 - 4 - 5 - 6 - 7 -

Note: There are specific endpoints to lock or unlock a device. The unlock actions differ slightly depending on the configuration of the door handle type - knob, handle or lever. Refer to the below table to understand the outcome of a simple lock action command.

Action	Outcome (based on the door handle type)		
	Smart Lock **Knob**	Smart Lock **Handle/Lever**	Opener
/lock	Lock	Lock	Deactivate rto and cm
/unlock	Unlatch (i.e opens door)	Unlock (i.e. doesn't open door)	Open

Refer to the minimum Firmware versions required to use the unlatch feature.

Nuki Device	Firmware Version
Bridge	1.14.0/2.5.0 (or higher)
Smart Lock 1.0	1.8.0 (or higher)
Smart Lock 2.0	2.4.3 (or higher)
Opener	1.3.0 (or higher)

N. Door State Changes

Door states are supported for Smart Locks with activated door sensors, from Smart Lock 2.0 onwards. The possible values for door state changes are:

- 0 unavailable
- 1 deactivated
- 2 door closed
- 3 door opened
- 4 door state unknown
- 5 calibrating

All door state changes of a Nuki device can be monitored by setting the Webhook feature "Device Status".

O. Error Codes

- Smart Lock Error Codes: https://developer.nuki.io/page/nuki-smart-lock-api-2/2/#heading--error-codes
- Opener Error Codes: https://developer.nuki.io/page/nuki-opener-api-1/7#heading--error-codes

HELP & SUPPORT

FAQs

1. What is the calling URL for Nuki Web API?
   https://api.nuki.io
2. What is Swagger?
   The Swagger UI is a tool we use to automatically generate documentation from our OpenAPI definition for visual interaction and easier testing for you.
3. How can I test the endpoints of the Nuki Web API?
   The Swagger interface allows you to easily try out the API commands from the interface. The Swagger JSON file can be downloaded at https://api.nuki.io/static/swagger/swagger.json or imported via Postman (File > Import > Link) to create a collection for it.
4. Where are the API parameters documented?
   Please look at the “Model” under the corresponding endpoint in Swagger for detailed documentation. Refer to the Section: Try the Demo for more information.
5. Why do all endpoints contain “Smartlock” instead of other device names?
   Endpoints containing "Smartlock" are used for all Nuki devices and are only kept that way for legacy reasons. All supported device types use the same endpoints. Only the Nuki Opener has additional ones for intercom compatibility.
6. What is OAuth 2?

OAuth 2 is an open standard for authentication. Nuki uses OAuth 2 to grant applications access to Nuki Web users' devices without sharing passwords.

7. What is the authorization bearer?
   We use API token as the authorization bearer for calls to Nuki Web API
8. How can I get a client secret for using the "Code Flow"?
   Refer to this section on "Apply for Nuki Web Advanced API" to obtain a client secret for using the OAuth2 Authorization Code Flow.
9. Can I use more than one redirect URL?
   You can add several redirect URLs as comma separated values.
10. Do you support the implicit flow of OAuth2?
    The implicit flow is supported on Swagger. The access token expires after one hour.
11. How long is the OAuth2 Code flow authorization valid?
    The access token expires in 1 hour, but the refresh token can be used to get a new access token. The refresh token may be invalidated when the authentication happens again with the same user account from a different device, application or session.
12. How can I convert my parameters in my requests to URL encoded parameters?
    Refer to the Online URL encoder/decoder.
13. How is unlatch different from unlock?
    The "Unlatch" feature just unlocks the door but doesn't open it. Refer to Definitions Section M: Smart Lock Actions to understand more about this.
14. Is there a limit on the maximum number of keypad codes created on a device?
    One Keypad can support up to 100 codes.
    Smart Locks 1st & 2nd Generations support up to 100 Keypad codes, and Smart Locks 3rd Generation onwards support up to 200 Keypad codes.
15. How can I differentiate between a Standard and Pro version?
    In the device configuration, if wifiEnabled = true, it means that the device's internal WiFi module can be used to bring the device online. This means it is a Pro version.
16. How can I resolve the CORS error?
    Cross-Origin Resource Sharing (CORS) is an HTTP mechanism that allows resource sharing from one origin to another origin securely. The CORS configuration changes on our API came into effect from March 2024 to comply with stricter security requirements. The access to API endpoints from localhost is no longer supported.
    One way to resolve this issue is through Proxying the Request. If you don't have control over the server's CORS configuration, you can proxy the request through your own server. Your server can make the request to 'https://api.nuki.io' on behalf of your web application. Since the request is coming from your server, CORS won't be an issue.
    For more information on this, refer to this link.

Abbreviations & Wordings

Wording	Abbreviation	Description
Authorization	auth	Any permission created to access a Nuki device (i.e.

		Nuki App user, keypad code, fob, fingerprint) Note: The authorization is stored on the device itself.
Continuous Mode	cm	Nuki Opener Mode with Ring to Open continuously activated
Keypad Code (access code)		6-digit code that is used to open a door from the Keypad connected to it Note: Keypad codes cannot contain ‘0’ and cannot start with ‘12’.
Lock 'n' Go	lng	Unlock and lock again automatically
Milliseconds	ms	One thousandth of a second
Ring to Open	rto	Nuki Opener State in which ringing the bell activates the electric strike actuation
Smartlock	SL	Refers to a Nuki device - Smart Lock, Opener, Smart Door or Box
Smart Lock ID	SL ID	The internal ID of a Nuki device

HTTP Status Codes

The following set of response codes may be returned when you send requests to the API.

Status Code	Description
200	Successful operation
204	Ok *(API request succeeded but action may or may not be successful - more info in the Best Practices section)*
400	Invalid E-Mail address or name supplied Email not in valid format One time password empty Bad Parameter Invalid parameter supplied Invalid parameter given
401	Not authorized Not authorized or one time password wrong
402	Account not payed
403	Forbidden

404	Not found Token not found
405	One time password is already enabled Not allowed
409	E-Mail address already exists Other account is already using the email Parameter conflicts
423	Locked
426	Account upgrade required
429	Too many failed attempts

Additional Resources

Nuki Developer Forum: https://developer.nuki.io/
Nuki Support: https://support.nuki.io/hc/en-us

CHANGE LOG

API Versions

v.1.5.3
13.11.2024

- Update the Web API documentation with information about Smart Lock Ultra

v.1.5.2
13.05.2024

- Added information on CORS in Best Practices and FAQs sections

v.1.5.1
14.02.2024

- Merged Web API Webhooks V1.1 documentation into Nuki Web API documentation
- Updated Nuki Web API documentation with use cases

v.1.5.0
31.08.2023

- Deprecated v.1.4.0

- API endpoints will return an error message if the Smart Lock or Smart Door doesn't have an active Smart Hosting subscription

v.1.4.1
21.06.2023

- Updated API endpoints to enforce the usage of the Nuki Web API (also referred to as Short Rental API) only with an active subscription for Nuki Smart Hosting

v.1.4.0
30.11.2021

- Added information on Smart Door and Smart Lock 3.0 (Pro) in the device ID section
- Added information on Smart Door and Smart Lock 3.0 (Pro) in the Smart Lock States

v.1.3.3
22.06.2021

- Added information on how to use the Swagger configuration in 3rd party tools

v.1.3.2
22.03.2021

- /notification endpoint (used for push notifications and webhooks) was deprecated

v.1.1.1 (Webhooks documentation)
02.02.2021

- Changed the message digest algorithm used to HMAC SHA256. To avoid issues with existing integrations a new header "X-Nuki-Signature-SHA256" has been added for that with the former one ("X-Nuki-Signature") remaining usable till the end of May 2021.

v.1.3.1
14.12.2020

- Minor text and formatting updates

v.1.3.0
02.03.2020

- Introduced Simple lock actions for all use cases where the logic should be handled by the device itself
- Added information on device ID usage
- Added Available endpoints to the Swagger part
- Added a description of the available Scopes to the API token section
- Added general naming conventions with Wording

v.1.2.1
14.01.2020

- Introduced the new section Advanced API integration to cover additional scopes which can only be accessed after registration and verification

v.1.2.0
31.05.2019

- Added support for the Nuki Opener to the Web API
- Added chapters for Smart Lock States and Actions to show differences between the Nuki Smart Lock and the Nuki Opener
- Noted changes and adding of new OpenerAdvancedSettings in section Swagger interface

v.1.1.1
30.08.2018

- Fixed some missing links
- Fixed some typos and unclear text