SnapKey

For all locks

SnapKey Public API

Documentation v1.3.0 — built 2026-09-19. The live version is at https://snapkey.dk/en/docs.

Getting started

What the API is for

The SnapKey Public API lets your own systems drive access control in SnapKey without anyone opening the dashboard. It carries two flows.

Your system → SnapKey. Your HR or facility system creates and updates the **people** who need access, and issues **keys** to them. Issuing a key sends the person a setup link (SMS or e-mail); the key issued this way appears once they activate that link. `GET /keys` also lists keys that never had a setup link — physical cards and iLOQ S5 fobs managed elsewhere — so read `type` to tell them apart.

SnapKey → your system. Read the access **events** from the doors (`GET /events`), or subscribe to **webhooks** and have SnapKey push each event to a URL you control.

Everything you can read or write is limited to the location your API key belongs to and the departments underneath it.

Get an API key

API keys are created in the SnapKey dashboard under **Developer → API keys**. The full token is shown **once**, at creation — store it in your secret manager; SnapKey keeps only a hash and the `sk_live_9f3c` prefix.

Every key carries an expiry date: **1 year** by default, up to **2 years** if the person creating it sets one further out. There is no non-expiring key.

The scopes a key holds are chosen when it is created. A request to an endpoint whose scope the key does not hold answers `403 insufficient_scope` .

Your first call

`GET /locks` lists the doors your key can see, and needs only the `catalog:read` scope — a good way to prove the token works.

curl

```
curl -X GET "https://api.snapkey.dk/public/v1/locks" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/locks", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```python
import os, requests

res = requests.get(
    "https://api.snapkey.dk/public/v1/locks",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```csharp
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.GetAsync("https://api.snapkey.dk/public/v1/locks");
var json = await res.Content.ReadAsStringAsync();
```

PHP

```php
$ch = curl_init("https://api.snapkey.dk/public/v1/locks");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "GET",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "data": [
    {
      "id": 4172,
      "name": "Main entrance",
      "serial_number": "S5-004172",
      "place": "Ground floor, east",
      "provider": "iloq",
      "online": null,
      "security_groups": [
        "HQ-STAFF"
      ],
      "location": {
        "id": 12,
        "name": "Headquarters"
      }
    }
  ],
  "next_cursor": null
}
```

Paging through a list

Every list endpoint returns a cursor page:

```
{ "data": [ … ], "next_cursor": "eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ" }
```

- `limit` — items per page, `1` – `200`, default `50`. Values outside the range are clamped.
- `cursor` — opaque; pass back the `next_cursor` of the previous page verbatim.

`next_cursor` is `null` on the last page. Rows are ordered by `id` ascending, so a page never reshuffles under you while you walk it.

curl

```
curl -X GET "https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.get(
    "https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.GetAsync("https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied");
var json = await res.Content.ReadAsStringAsync();
```

PHP

```php
$ch = curl_init("https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "GET",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```

Walking every page means repeating the same request with the `next_cursor` you were just handed, until `next_cursor` comes back `null`:

```
GET /public/v1/events?limit=50
GET /public/v1/events?limit=50&cursor=<next_cursor from the previous page>
```

Send SnapKey a person and issue a key

First create the person. `POST /people` is idempotent on phone and e-mail: if a person in the API key's scope already has the same normalised phone number or the same e-mail address, that person is returned with **200 OK** and nothing is changed. A genuinely new person answers **201 Created**.

curl

```curl
curl -X POST "https://api.snapkey.dk/public/v1/people" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "name": "Mette Sørensen", "email": "mette.sorensen@example.com", "phone":
        "+4520123456", "company_name": "Tidevand Energi", "title": "Facility Manager",
        "language": "da", "location_id": 12 }'
```

Node

```node
const res = await fetch("https://api.snapkey.dk/public/v1/people", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "phone": "+4520123456",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location_id": 12
  }),
```

```
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/people",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
    json={
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "phone": "+4520123456",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location_id": 12
},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var content = new StringContent("{ \"name\": \"Mette Sørensen\", \"email\":
        \"mette.sorensen@example.com\", \"phone\": \"+4520123456\", \"company_name\":
        \"Tidevand Energi\", \"title\": \"Facility Manager\", \"language\": \"da\",
        \"location_id\": 12 }", Encoding.UTF8, "application/json");
var res = await http.PostAsync("https://api.snapkey.dk/public/v1/people", content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/people");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["name" => "Mette Sørensen", "email" =>
        "mette.sorensen@example.com", "phone" => "+4520123456", "company_name" =>
        "Tidevand Energi", "title" => "Facility Manager", "language" => "da",
        "location_id" => 12]),
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
```

```
  "name": "Mette Sørensen",
  "email": "mette.sorensen@example.com",
  "phone": "+4520123456",
  "company_name": "Tidevand Energi",
  "title": "Facility Manager",
  "language": "da",
  "location": {
    "id": 12,
    "name": "Headquarters"
  }
}
```

Then issue the key with the person's `id`. SnapKey creates an **invitation** and sends the person a setup link over SMS, e-mail or both; the call answers **202 Accepted** with the invitation, not a key.

curl

```
curl -X POST "https://api.snapkey.dk/public/v1/keys" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -d '{ "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40", "security_groups": [ "HQ-
        STAFF" ], "name": "Headquarters staff", "starts_at": "2026-09-01T00:00:00Z",
        "expires_at": "2027-09-01T00:00:00Z", "channel": "sms" }'
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/keys", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "security_groups": [
      "HQ-STAFF"
    ],
    "name": "Headquarters staff",
    "starts_at": "2026-09-01T00:00:00Z",
    "expires_at": "2027-09-01T00:00:00Z",
    "channel": "sms"
  }),
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/keys",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
    json={
```

```
    "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "security_groups": [
        "HQ-STAFF"
    ],
    "name": "Headquarters staff",
    "starts_at": "2026-09-01T00:00:00Z",
    "expires_at": "2027-09-01T00:00:00Z",
    "channel": "sms"
},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var content = new StringContent("{ \"person_id\": \"9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40\", \"security_groups\": [ \"HQ-STAFF\" ], \"name\": \"Headquarters
        staff\", \"starts_at\": \"2026-09-01T00:00:00Z\", \"expires_at\": \"2027-09-
        01T00:00:00Z\", \"channel\": \"sms\" }", Encoding.UTF8, "application/json");
var res = await http.PostAsync("https://api.snapkey.dk/public/v1/keys", content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/keys");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["person_id" => "9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40", "security_groups" => ["HQ-STAFF"], "name" => "Headquarters
        staff", "starts_at" => "2026-09-01T00:00:00Z", "expires_at" => "2027-09-
        01T00:00:00Z", "channel" => "sms"]),
]);
$data = json_decode(curl_exec($ch), true);
```

The key itself appears in `GET /keys` with state `handed_over` once the person activates the setup link, and fires the `key.activated` webhook at that moment.

Manage a key after issuing it

The `invitation_id` from `POST /keys` is a resource of its own. Read it to see whether the person has activated the link yet:

curl

```
curl -X GET "https://api.snapkey.dk/public/v1/invitations/<id>" \
```

```
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/invitations/<id>", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.get(
    "https://api.snapkey.dk/public/v1/invitations/<id>",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.GetAsync("https://api.snapkey.dk/public/v1/invitations/<id>");
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/invitations/<id>");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "GET",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "id": 5521,
  "state": "activated",
  "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
  "key_id": 88213,
  "name": "Headquarters staff",
  "security_groups": [
    "HQ-STAFF"
  ],
```

```
  "starts_at": "2026-09-01T00:00:00Z",
  "expires_at": "2027-09-01T00:00:00Z",
  "sent": {
    "email_at": null,
    "sms_at": "2026-09-19T08:12:03Z"
  },
  "created_at": "2026-09-19T08:12:02Z"
}
```

`state` moves from `pending` to `activated` when the person turns the link into a key; `key_id` then names that key. If the message never reached them, send it again — over SMS, e-mail or both:

curl

```
curl -X POST "https://api.snapkey.dk/public/v1/invitations/<id>/resend" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "channel": "both" }'
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/invitations/<id>/resend", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "channel": "both"
  }),
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/invitations/<id>/resend",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
    json={
    "channel": "both"
},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
```

```
var content = new StringContent("{ \"channel\": \"both\" }", Encoding.UTF8,
        "application/json");
var res = await
        http.PostAsync("https://api.snapkey.dk/public/v1/invitations/<id>/resend",
        content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/invitations/<id>/resend");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["channel" => "both"]),
]);
$data = json_decode(curl_exec($ch), true);
```

A link sent to the wrong person is recalled with `DELETE /invitations/{id}`; it cannot be activated afterwards. Both calls work on `pending` invitations only and answer `409 invitation_not_pending` otherwise.

Once the key exists, change its security groups or validity window in place:

curl

```
curl -X PATCH "https://api.snapkey.dk/public/v1/keys/<id>" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "security_groups": [ "HQ-STAFF", "HQ-PARKING" ], "expires_at": "2027-09-
        01T00:00:00Z" }'
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/keys/<id>", {
  method: "PATCH",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "security_groups": [
      "HQ-STAFF",
      "HQ-PARKING"
    ],
    "expires_at": "2027-09-01T00:00:00Z"
  }),
});
const json = await res.json();
```

Python

```python
import os, requests

res = requests.patch(
    "https://api.snapkey.dk/public/v1/keys/<id>",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
    json={
    "security_groups": [
        "HQ-STAFF",
        "HQ-PARKING"
    ],
    "expires_at": "2027-09-01T00:00:00Z"
},
)
data = res.json()
```

C#

```csharp
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var content = new StringContent("{ \"security_groups\": [ \"HQ-STAFF\", \"HQ-PARKING\"
        ], \"expires_at\": \"2027-09-01T00:00:00Z\" }", Encoding.UTF8,
        "application/json");
var res = await http.PatchAsync("https://api.snapkey.dk/public/v1/keys/<id>", content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```php
$ch = curl_init("https://api.snapkey.dk/public/v1/keys/<id>");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "PATCH",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["security_groups" => ["HQ-STAFF", "HQ-PARKING"],
        "expires_at" => "2027-09-01T00:00:00Z"]),
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "id": 88213,
  "name": "Headquarters staff",
  "type": "digital",
  "state": "sent",
  "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
  "invitation_id": 5521,
  "security_groups": [
    "HQ-STAFF",
    "HQ-PARKING"
```

```
  ],
  "starts_at": "2026-09-01T00:00:00Z",
  "expires_at": "2027-09-01T00:00:00Z"
}
```

Any change other than a rename puts the key back into state `sent` until the locking system confirms it, and `key.activated` fires again at that moment. Revoking is still `DELETE /keys/{id}`.

Live status and remote unlock

A lock SnapKey opens over the network reports whether its device is reachable:

curl

```
curl -X GET "https://api.snapkey.dk/public/v1/locks/<id>" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/locks/<id>", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.get(
    "https://api.snapkey.dk/public/v1/locks/<id>",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.GetAsync("https://api.snapkey.dk/public/v1/locks/<id>");
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/locks/<id>");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "GET",
```

```
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "id": 4172,
  "name": "Main entrance",
  "serial_number": null,
  "place": "Ground floor, east",
  "provider": "teltonika",
  "online": true,
  "security_groups": [
    "HQ-STAFF"
  ],
  "location": {
    "id": 12,
    "name": "Headquarters"
  }
}
```


`online` is `null` for an iLOQ lock — SnapKey cannot see a cylinder. To open a remote lock your API key needs the `locks:control` scope **and** the lock on its allowlist, chosen when the key is created in the dashboard. The call answers `202` with a command; the door is not open yet.

curl

```
curl -X POST "https://api.snapkey.dk/public/v1/locks/<id>/unlock" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/locks/<id>/unlock", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/locks/<id>/unlock",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
```

```
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.PostAsync("https://api.snapkey.dk/public/v1/locks/<id>/unlock",
        null);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/locks/<id>/unlock");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```

```
{
  "id": "0d3f6c2a-7b1e-4f0a-9c8d-2e5b7a1f4c33",
  "lock_id": 4172,
  "status": "published",
  "requested_at": "2026-09-19T09:40:11Z",
  "published_at": "2026-09-19T09:40:11Z",
  "confirmed_at": null,
  "timeout_seconds": 10,
  "error": null
}
```

Poll the command, or wait for `access.granted` (the device confirmed) or `unlock.failed` (it did not) on your webhook. A lock outside the key's allowlist answers `403 lock_not_allowed` and is logged as a refused access, so the attempt shows up in `GET /events` like any other.

Get live feedback

Subscribe a URL to the door and key events and SnapKey pushes each one to you as it is recorded — see Webhooks for the payloads, the signature check and the retry ladder.

`GET /events` serves the same door events as a list: poll it, and replay from it with `since=` after a webhook gap.

Tools

Import the OpenAPI file into Postman or Insomnia:

```
https://api.snapkey.dk/docs/openapi.json
```

Prefer a document? Download the full documentation as PDF from the box below.

Authentication & limits

Bearer token

Every request carries a bearer token:

```
Authorization: Bearer sk_live_9f3c…
```

API keys are created in the SnapKey dashboard under **Developer** → **API keys**. The full token is shown **once**, at creation — store it in your secret manager; SnapKey keeps only a hash and the `sk_live_9f3c` prefix. A key can be revoked at any time from the same screen; a revoked or expired key answers `401 unauthenticated`.

Scopes

A key carries an explicit list of scopes. A request to an endpoint whose scope the key does not hold answers `403 insufficient_scope`.

Scope	Grants
`catalog:read`	`GET /locks`, `GET /locks/{id}`, `GET /security_groups`
`people:read`	`GET /people`, `GET /people/{id}`
`people:write`	`POST`, `PUT`, `DELETE` on `/people` — and everything `people:read` grants
`keys:read`	`GET /keys`, `GET /keys/{id}`, `GET /invitations`, `GET /invitations/{id}`
`keys:write`	`POST /keys`, `PATCH`, `DELETE` on `/keys/{id}`, `POST /invitations/{id}/resend`, `DELETE /invitations/{id}` — and everything `keys:read` grants
`locks:control`	`POST /locks/{id}/unlock`, `GET /locks/{id}/commands/{command_id}` — only for the locks listed on the key
`events:read`	`GET /events`
`webhooks:manage`	all `/webhooks` endpoints

A read endpoint accepts either scope: `GET /people` is served for a key holding `people:read` **or** `people:write`.

What a key can see

An API key belongs to exactly one location. Every list, lookup and write is limited to **that location and its descendants** — never the account root, never a sibling department. A row that exists elsewhere in the account is not "forbidden", it is simply `404 not_found`.

Reseller (platform-mode) accounts are not supported in v1: their keys answer `403 not_available` on every endpoint.

Key expiry and rotation

Every key carries an expiry date: **1 year** by default, up to **2 years** if the person creating it sets one further out. There is no non-expiring key — plan to rotate before `expires_at`, which the API keys screen in the dashboard shows for every key.

Rotate by creating the next key in the dashboard, moving your secret manager over to it, and then revoking the old one.

Rate limits

600 requests per minute per API key. A request that arrives without a bearer token is counted against a shared per-IP bucket of the same size instead.

Every response from an **authenticated** endpoint carries:

Header	Meaning
`X-RateLimit-Limit`	requests allowed per minute (600)
`X-RateLimit-Remaining`	requests left in the current window

Exceeding a limit answers `429 rate_limited` with a `Retry-After` header holding the number of seconds to wait. Two endpoints are limited harder, because each one makes SnapKey act on the outside world:

Endpoint	Limit
`POST /keys`	30 per minute, per API key — every call sends an SMS or e-mail
`POST /webhooks/{id}/ping`	10 per minute, per API key

On `POST /keys`, `X-RateLimit-Limit` shows the limiter closest to exhaustion on that route, so it can show `30` rather than the account-wide 600. The `POST /keys` limit is counted before validation and scope checks — a rejected request still consumes one.

Requests that fail to authenticate are counted separately: **30 failed authentications per minute, per IP**. Once an IP has produced 30 failed authentications in a minute, every request from that IP is answered `429` until the window rolls — including requests with a valid key. `Retry-After` tells you how long. If you share an egress IP with other tenants or with misconfigured clients, expect this and back off.

Idempotency

`POST /people`, `POST /keys` and `POST /webhooks` accept an optional `Idempotency-Key` request header — 1–64 characters of `A–Z a–z 0–9 _ –`. Send one per logical operation and reuse it when you retry after a timeout.

- The first request runs normally; its status and body are stored for **24 hours**, per API key.
- A retry with the same key **and the same body** returns the stored response verbatim, with `Idempotent-Replayed: true`. That is the answer to "did my retry create a duplicate?": no.

- The same key with a **different** body answers `409 idempotency_conflict`.
- A retry that arrives while the first call is still running answers `409 idempotency_in_progress` with `Retry-After: 2`.
- `5xx` responses and `409 iloq_rejected` are never stored — both are transient — so a retry with the same key re-executes.
- Responses larger than 64 KB, or that are not JSON, are not stored: a retry with the same key executes the request again (never a replay).
- A key that is still in progress after **60 seconds** is treated as abandoned (a killed worker, a dropped connection) and the retry re-executes rather than waiting out the 24 hours.

Without the header nothing changes: `POST /people` still de-duplicates on phone and e-mail, and `POST /keys` still sends every time.

curl

```
curl -X POST "https://api.snapkey.dk/public/v1/people" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "name": "Mette Sørensen", "email": "mette.sorensen@example.com", "phone":
        "+4520123456", "company_name": "Tidevand Energi", "title": "Facility Manager",
        "language": "da", "location_id": 12 }'
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/people", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "phone": "+4520123456",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location_id": 12
  }),
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/people",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
    json={
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
```

```
    "phone": "+4520123456",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location_id": 12
},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var content = new StringContent("{ \"name\": \"Mette Sørensen\", \"email\":
        \"mette.sorensen@example.com\", \"phone\": \"+4520123456\", \"company_name\":
        \"Tidevand Energi\", \"title\": \"Facility Manager\", \"language\": \"da\",
        \"location_id\": 12 }", Encoding.UTF8, "application/json");
var res = await http.PostAsync("https://api.snapkey.dk/public/v1/people", content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/people");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["name" => "Mette Sørensen", "email" =>
        "mette.sorensen@example.com", "phone" => "+4520123456", "company_name" =>
        "Tidevand Energi", "title" => "Facility Manager", "language" => "da",
        "location_id" => 12]),
]);
$data = json_decode(curl_exec($ch), true);
```

Add `Idempotency-Key: <uuid>` to this request and reuse it on retry:

```
POST /public/v1/people
Idempotency-Key: 3f1c7a52-9b40-4f6d-8b2e-0a7c5d1e9f84
```

Webhooks

Why webhooks

A subscription has SnapKey push each event to a URL you control, so your system is told as soon as SnapKey records it instead of polling for it. You get the door events — `access.granted`, `access.denied`, `door.closed`, `door.left_open` — and the key lifecycle: `key.activated` when a person activates a setup link, `key.revoked` when a key is returned or deleted.

`GET /events` stays the endpoint for everything a push cannot do: backfilling history, replaying the gap after a paused subscription, and reading the access log for an audit.

Create a subscription

Subscribe a URL to one or more event types. Pass `"*"` to receive every type.

curl

```
curl -X POST "https://api.snapkey.dk/public/v1/webhooks" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "url": "https://hooks.example.com/snapkey", "events": [ "access.granted",
        "access.denied" ], "description": "Tidevand Energi facility dashboard" }'
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/webhooks", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "url": "https://hooks.example.com/snapkey",
    "events": [
      "access.granted",
      "access.denied"
    ],
    "description": "Tidevand Energi facility dashboard"
  }),
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/webhooks",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
```

```
    json={
    "url": "https://hooks.example.com/snapkey",
    "events": [
        "access.granted",
        "access.denied"
    ],
    "description": "Tidevand Energi facility dashboard"
},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var content = new StringContent("{ \"url\": \"https://hooks.example.com/snapkey\",
        \"events\": [ \"access.granted\", \"access.denied\" ], \"description\":
        \"Tidevand Energi facility dashboard\" }", Encoding.UTF8, "application/json");
var res = await http.PostAsync("https://api.snapkey.dk/public/v1/webhooks", content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/webhooks");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["url" => "https://hooks.example.com/snapkey",
        "events" => ["access.granted", "access.denied"], "description" => "Tidevand
        Energi facility dashboard"]),
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
  "url": "https://hooks.example.com/snapkey",
  "events": [
    "access.granted",
    "access.denied"
  ],
  "description": "Tidevand Energi facility dashboard",
  "status": "active",
  "failing_since": null,
  "last_delivery_at": null,
  "created_at": "2026-09-02T09:12:00Z",
  "secret": "whsec_3f8c1d90a47b4e2fa6c5029d81be7a34"
}
```

The response carries the signing `secret` **once** — store it now; it is never returned again. `GET /webhooks` never returns it. The secret is always generated by SnapKey; a `secret` field in the request is rejected as an unknown field.

The `url` must be `https://` and resolve to a public host; loopback, private-range and `.local` / `.internal` hosts are refused. The API key's own location may hold at most 5 active subscriptions — the sixth answers `422` with detail code `limit_reached`. The cap is per location, so departments under it have their own allowance.

You can also create a subscription in the SnapKey dashboard, under **Developer → Webhooks**.

Manage a subscription from your system

`GET /webhooks/{id}` returns one subscription and `PATCH /webhooks/{id}` changes it, so you never have to open the dashboard to keep an integration running. You can change the `url`, the `events` list, the `description` and the `status`. At least one of them has to be in the body — an empty one is refused with `422 validation_failed`.

curl

```
curl -X PATCH "https://api.snapkey.dk/public/v1/webhooks/<id>" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "status": "active" }'
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/webhooks/<id>", {
  method: "PATCH",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    "status": "active"
  }),
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.patch(
    "https://api.snapkey.dk/public/v1/webhooks/<id>",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
    json={
    "status": "active"
},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var content = new StringContent("{ \"status\": \"active\" }", Encoding.UTF8,
        "application/json");
var res = await http.PatchAsync("https://api.snapkey.dk/public/v1/webhooks/<id>",
        content);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/webhooks/<id>");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "PATCH",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
        "Content-Type: application/json",
    ],
    CURLOPT_POSTFIELDS => json_encode(["status" => "active"]),
]);
$data = json_decode(curl_exec($ch), true);
```

```
{
  "id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
  "url": "https://hooks.example.com/snapkey",
  "events": [
    "access.granted",
    "access.denied"
  ],
  "description": "Tidevand Energi facility dashboard",
  "status": "active",
  "failing_since": null,
  "last_delivery_at": "2026-09-02T08:15:31Z",
  "created_at": "2026-08-20T09:00:00Z"
}
```

A new `url` is validated exactly like the one you created the subscription with: `https://`, a public host, at most 2048 characters. Changing the URL does not rotate the signing secret. A new `events` list is validated like the one on create — `"*"` or known event names, at least one.

`status` takes `paused` or `active`, and nothing else. `paused` stops deliveries: nothing new is queued and retries still in flight fail. `active` on a paused subscription resumes it and clears the failure streak, exactly what **Resume** in the dashboard does. `active` on a subscription that is already active changes nothing.

Events

Event	Meaning
`access.granted`	A door was opened
`access.denied`	A door refused a key
`door.closed`	A door was closed
`door.left_open`	A door did not report back
`key.activated`	A key was activated
`key.revoked`	A key was revoked
`key.issued`	A key invitation was sent
`unlock.failed`	A remote unlock was never confirmed by the device
`lock.online`	A remote lock became reachable
`lock.offline`	A remote lock stopped being reachable
`person.created`	A person was created
`person.updated`	A person's details changed
`person.deleted`	A person was deleted
`ping`	A test delivery

access.granted

Sent when a key opened a lock. `data` is the same object `GET /events` returns.


```
{
  "id": "6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55",
  "type": "access.granted",
  "created_at": "2026-09-02T08:15:31Z",
  "data": {
    "id": 9910427,
    "type": "access.granted",
    "occurred_at": "2026-09-02T08:15:30Z",
    "result": "success",
    "reason": null,
    "source": "iloq",
    "api_key": null,
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
      "name": "Headquarters"
```

```
    },
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "key": {
      "id": 88213,
      "name": "Headquarters staff"
    }
  }
}
```


access.denied

Sent when a lock refused a key. `data.reason` says why: `no_access`, `outside_time_window`, `unknown_key`, or `null` when the locking system gave no reason. A refusal is the one case that can arrive with no `person` and no `key` — an unknown key was presented, as below.


```
{
  "id": "2b7d40e9-9c11-4f6a-b3d5-1e08a7c62f93",
  "type": "access.denied",
  "created_at": "2026-09-02T19:42:12Z",
  "data": {
    "id": 9910435,
    "type": "access.denied",
    "occurred_at": "2026-09-02T19:42:11Z",
    "result": "denied",
    "reason": "unknown_key",
    "source": "iloq",
    "api_key": null,
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
      "name": "Headquarters"
    },
    "person": null,
    "key": null
  }
}
```


door.closed

Sent when a lock reported the door closed. The event inherits `person` and `key` from the access that opened the door.


```
{
  "id": "8a5c3e10-64b2-4d8f-a09c-2f71b4e5d602",
  "type": "door.closed",
```

```
  "created_at": "2026-09-02T08:16:02Z",
  "data": {
    "id": 9910429,
    "type": "door.closed",
    "occurred_at": "2026-09-02T08:16:01Z",
    "result": "success",
    "reason": null,
    "source": "iloq",
    "api_key": null,
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
      "name": "Headquarters"
    },
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "key": {
      "id": 88213,
      "name": "Headquarters staff"
    }
  }
}
```


door.left_open

Sent when a lock did not report a close after being opened — the door is standing open, or the lock stopped answering. The event inherits `person` and `key` from the access that opened the door.


```
{
  "id": "1c4f7b28-05de-4a63-91b7-c8e02a6d3f45",
  "type": "door.left_open",
  "created_at": "2026-09-02T08:21:10Z",
  "data": {
    "id": 9910431,
    "type": "door.left_open",
    "occurred_at": "2026-09-02T08:21:09Z",
    "result": "success",
    "reason": null,
    "source": "iloq",
    "api_key": null,
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
```

```
      "name": "Headquarters"
    },
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "key": {
      "id": 88213,
      "name": "Headquarters staff"
    }
  }
}
```


key.activated

Sent whenever a key's state becomes `handed_over` — a person activating a setup link, a manager handing a key over in the SnapKey dashboard, or a reconciliation against the locking system. It is **not** a receipt for one of your `POST /keys` calls, so reconcile on `data.key.id` (which is also `data.id`), never on your own invitation ids.


```
{
  "id": "5e9b6f31-7a24-4c08-8de1-30b95c2f7a86",
  "type": "key.activated",
  "created_at": "2026-09-01T07:03:22Z",
  "data": {
    "id": 88213,
    "type": "key.activated",
    "occurred_at": "2026-09-01T07:03:22Z",
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "key": {
      "id": 88213,
      "name": "Headquarters staff",
      "state": "handed_over"
    }
  }
}
```


key.revoked

Sent when a key is returned or deleted — including by `DELETE /keys/{id}`.


```
{
  "id": "7d31c8a0-4e6b-4915-b2fa-91c07de4a538",
  "type": "key.revoked",
  "created_at": "2026-09-02T10:30:00Z",
  "data": {
    "id": 88213,
    "type": "key.revoked",
```

```
    "occurred_at": "2026-09-02T10:30:00Z",
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "key": {
      "id": 88213,
      "name": "Headquarters staff",
      "state": "returned"
    }
  }
}
```


key.issued

Sent when an invitation for a key has been sent — by `POST /keys` or from the dashboard. `channel` says how it went out: `sms`, `email` or `both`. This is the invitation, not the key coming into use; `key.activated` follows when the recipient activates it.


```
{
  "id": "1a9d4e70-6c2b-4f18-9a3e-8b05c7d21f64",
  "type": "key.issued",
  "created_at": "2026-09-02T09:00:00Z",
  "data": {
    "invitation_id": 55211,
    "channel": "sms",
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "security_groups": [
      "HQ-STAFF"
    ],
    "starts_at": "2026-09-02T09:00:00Z",
    "expires_at": "2026-12-02T09:00:00Z"
  }
}
```


unlock.failed

Sent when a remote unlock was ordered but the device never confirmed it. `data` is the access-event object with a `reason` alongside it: `timed_out`, `device_missing`, `publish_failed` or `device_error`. This one is webhook-only — a remote unlock that was never confirmed is not a door event, so it never appears in `GET /events`.


```
{
  "id": "3f8a6c12-9e04-4b7d-a1c6-5d20e8f4b937",
  "type": "unlock.failed",
  "created_at": "2026-09-05T14:02:11Z",
  "data": {
    "id": 9911042,
```

```
    "type": "unlock.failed",
    "occurred_at": "2026-09-05T14:02:10Z",
    "result": "denied",
    "reason": "timed_out",
    "source": "app",
    "api_key": null,
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
      "name": "Headquarters"
    },
    "person": {
      "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
      "name": "Mette Sørensen",
      "email": "mette.sorensen@example.com"
    },
    "key": null
  }
}
```


lock.online

Sent when a remote lock becomes reachable again. Presence is reported per gateway and the event is sent once for every lock on it, with `since` giving the moment the state changed.


```
{
  "id": "2d6b9f04-5a13-4c8e-8f70-1b94d6e5c832",
  "type": "lock.online",
  "created_at": "2026-09-05T10:00:05Z",
  "data": {
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
      "name": "Headquarters"
    },
    "online": true,
    "since": "2026-09-05T10:00:02Z"
  }
}
```


lock.offline

Sent when a remote lock stops being reachable. A disconnect is reported within seconds of the unit dropping; a unit that simply goes quiet is reported offline after 45 minutes without a heartbeat. As

with `lock.online`, one event is sent per lock on the gateway, and `since` gives the moment the state changed.


```
{
  "id": "8e1c4a70-3f26-4d9b-9a05-6c82e7f10b53",
  "type": "lock.offline",
  "created_at": "2026-09-05T11:15:03Z",
  "data": {
    "lock": {
      "id": 4172,
      "name": "Main entrance",
      "place": "Ground floor, east"
    },
    "location": {
      "id": 12,
      "name": "Headquarters"
    },
    "online": false,
    "since": "2026-09-05T10:30:00Z"
  }
}
```


person.created

Sent when a person is created, whatever created them — this API, the dashboard, SCIM provisioning or the iLOQ sync. `data` is the same object `GET /people/{id}` returns, so it holds only the fields the API already shows you.


```
{
  "id": "7c2e9a41-3d68-4b0f-9c17-2a85e6d40b73",
  "type": "person.created",
  "created_at": "2026-09-02T09:05:00Z",
  "data": {
    "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "phone": "+4520123456",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location": {
      "id": 12,
      "name": "Headquarters"
    }
  }
}
```


person.updated

Sent when one of a person's client-visible fields changes — name, e-mail, phone, company, title or language. `changed_fields` lists which ones. A save that touches only internal columns sends nothing.

```
{
  "id": "5b1f8d63-2a47-4e9c-8b06-3f71c9a52e08",
  "type": "person.updated",
  "created_at": "2026-09-02T09:10:00Z",
  "data": {
    "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "phone": "+4520987654",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location": {
      "id": 12,
      "name": "Headquarters"
    },
    "changed_fields": [
      "phone"
    ]
  }
}
```


person.deleted

Sent when a person is deleted, soft or hard.


```
{
  "id": "4e0a7c25-8b31-4f6d-9c02-7a15e3f8b904",
  "type": "person.deleted",
  "created_at": "2026-09-02T09:15:00Z",
  "data": {
    "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "location": {
      "id": 12,
      "name": "Headquarters"
    }
  }
}
```


ping

Sent by `POST /webhooks/{id}/ping`. Never produced by a real event, and never subscribed to — use it to confirm your endpoint is reachable and your signature check works.


```
{
  "id": "6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55",
  "type": "ping",
  "created_at": "2026-09-02T09:13:03Z",
  "data": {
    "message": "pong"
```

```
  }
}
```

What a delivery looks like

A subscription receives an HTTP `POST` for every matching event:

```
POST https://hooks.example.com/snapkey
Content-Type: application/json
User-Agent: SnapKey-Webhooks/1.0
X-SnapKey-Event: access.granted
X-SnapKey-Delivery: 6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55
X-SnapKey-Timestamp: 1788336930
X-SnapKey-Signature: t=<unix seconds>,v1=<hex HMAC-SHA256(secret, "{t}.{raw body}")>

{ "id": "6f0a1d5c-…", "type": "access.granted", "created_at": "2026-09-02T08:15:30Z",
"data": { … } }
```

Header	Meaning
`X-SnapKey-Event`	The event type, identical to `type` in the body.
`X-SnapKey-Delivery`	The delivery UUID, identical to `id` in the body. Stable across retries.
`X-SnapKey-Timestamp`	Unix seconds at which this attempt was sent.
`X-SnapKey-Signature`	`t=<unix seconds>,v1=<hex HMAC-SHA256(secret, "{t}.{raw body}")>`
`User-Agent`	`SnapKey-Webhooks/1.0`

The envelope is the same for every type:

- `id` — the delivery id, stable across retries.
- `type` — the event type.
- `created_at` — when SnapKey created the delivery, not when this attempt was sent.
- `data` — the event itself. For a door event it is the same object `GET /events` returns.

Verify the signature

1. Split `X-SnapKey-Signature` on `,` into `t=<unix seconds>` and `v1=<hex digest>`.
2. Reject the delivery if `t` is more than **300 seconds** away from your own clock — that is what stops a captured delivery from being replayed later.
3. Compute `HMAC-SHA256` over the string `"{t}.{raw body}"` — the raw bytes you received, not a re-serialised copy — keyed with your subscription secret.
4. Compare it with `v1` in **constant time** (`hash_equals` , `crypto.timingSafeEqual` , …).

Node

```
import { createHmac, timingSafeEqual } from "node:crypto";
```

```
export function verifySnapKeySignature(header, rawBody, secret, now = Date.now() /
        1000) {
  const parts = Object.fromEntries(header.split(",").map((p) => p.split("=")));
  const t = Number(parts.t);
  if (!Number.isFinite(t) || Math.abs(now - t) > 300) return false;
  const expected = createHmac("sha256",
        secret).update(`${t}.${rawBody}`).digest("hex");
  const given = String(parts.v1 ?? "");
  return expected.length === given.length && timingSafeEqual(Buffer.from(expected),
        Buffer.from(given));
}
```

Python

```
from __future__ import annotations

import hmac, hashlib, time

def verify_snapkey_signature(header: str, raw_body: bytes, secret: str, now: float |
        None = None) -> bool:
    parts = dict(p.split("=", 1) for p in header.split(","))
    if "t" not in parts:
        return False
    try:
        t = int(parts["t"])
    except ValueError:
        return False
    if abs((now or time.time()) - t) > 300:
        return False
    expected = hmac.new(secret.encode(), f"{t}.".encode() + raw_body,
        hashlib.sha256).hexdigest()
    return hmac.compare_digest(expected, parts.get("v1", ""))
```

C#

```
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;

static bool VerifySnapKeySignature(string header, string rawBody, string secret, long?
        now = null)
{
    var parts = header.Split(',').Select(p => p.Split('=', 2)).ToDictionary(p => p[0],
        p => p[1]);
    if (!(parts.TryGetValue("t", out var tRaw) && long.TryParse(tRaw, out var t)))
        return false;
    var unixNow = now ?? DateTimeOffset.UtcNow.ToUnixTimeSeconds();
    if (Math.Abs(unixNow - t) > 300) return false;
    using var hmac = new HMACSHA256(Encoding.UTF8.GetBytes(secret));
    var expected = Convert.ToHexString(hmac.ComputeHash(Encoding.UTF8.GetBytes($"{t}.
        {rawBody}"))).ToLowerInvariant();
    return CryptographicOperations.FixedTimeEquals(Encoding.UTF8.GetBytes(expected),
        Encoding.UTF8.GetBytes(parts.GetValueOrDefault("v1", "")));
}
```

PHP

```php
function verifySnapKeySignature(string $header, string $rawBody, string $secret, ?int
        $now = null): bool
{
    [$t, $v1] = sscanf($header, 't=%d,v1=%s');
    if (abs(($now ?? time()) - $t) > 300) {
        return false;
    }
    return hash_equals(hash_hmac('sha256', $t.'.'.$rawBody, $secret), (string) $v1);
}
```

Respond fast, expect retries

Respond with any 2xx within 10 seconds. Anything else — a non-2xx status, a timeout, a connection failure — is a failed attempt. Redirects are not followed; any 3xx is a failed attempt. SnapKey retries after **1 m, 5 m, 30 m, 2 h, 6 h, 24 h**, then marks the delivery `exhausted` and stops.

Paused subscriptions

A subscription that has been failing continuously for 24 hours is set to `paused`, and the user who created the API key (or the subscription) is e-mailed, when one is known. A paused subscription receives nothing; resume it from the dashboard or with `PATCH /webhooks/{id}`, and recover the gap with `GET /events?since=`.

curl

```curl
curl -X GET "https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```node
const res = await fetch("https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```python
import os, requests

res = requests.get(
```

```
    "https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.GetAsync("https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied");
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch = curl_init("https://api.snapkey.dk/public/v1/events?since=2026-09-
        01T00%3A00%3A00Z&lock_id=4172&person_id=9c1f2a84-3b7e-4d21-9f60-
        5a2c8d7e1b40&type=access.granted%2Caccess.denied");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "GET",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```

Deliveries and redelivery

`GET /webhooks/{id}/deliveries` lists what SnapKey tried to send to a subscription, newest first and cursor paged, so you can see exactly which attempts your endpoint accepted or rejected.

curl

```
curl -X GET "https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries?
        event_type=access.granted&since=2026-09-01T00%3A00%3A00Z" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries?
        event_type=access.granted&since=2026-09-01T00%3A00%3A00Z", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```python
import os, requests

res = requests.get(
    "https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries?
        event_type=access.granted&since=2026-09-01T00%3A00%3A00Z",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```csharp
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await
        http.GetAsync("https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries?
        event_type=access.granted&since=2026-09-01T00%3A00%3A00Z");
var json = await res.Content.ReadAsStringAsync();
```

PHP

```php
$ch = curl_init("https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries?
         event_type=access.granted&since=2026-09-01T00%3A00%3A00Z");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "GET",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```


```
{
  "data": [
    {
      "id": "6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55",
      "subscription_id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
      "event_type": "access.granted",
      "event_id": "9910427",
      "status": "delivered",
      "attempt": 1,
      "response_status": 200,
      "last_error": null,
      "next_attempt_at": null,
      "delivered_at": "2026-09-02T08:15:31Z",
      "created_at": "2026-09-02T08:15:30Z",
      "payload": {
        "id": 9910427,
        "type": "access.granted",
        "occurred_at": "2026-09-02T08:15:30Z"
      }
```

```
    }
  ],
  "next_cursor": null
}
```

Narrow the list with `status` (a delivery status), `event_type` (one event name) and `since` (an ISO 8601 timestamp).

`POST /webhooks/{id}/deliveries/{delivery_id}/redeliver` sends one of them again. It creates a fresh delivery carrying the same event type and payload, with the original delivery's id as its `event_id`, so the duplicate guard never swallows it. The original row is left untouched. The new delivery starts over and follows the full retry ladder, and any delivery can be redelivered whatever its status — including ones that were delivered successfully, and pings.

Redelivery only works on an active subscription: a paused one answers `409` with the code `subscription_paused`. Resume the subscription first. Redelivery is limited to 30 calls per minute per API key.

curl

```
curl -X POST
        "https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries/<deliveryId>/redeliv
        er" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await
        fetch("https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries/<deliveryId>/r
        edeliver", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries/<deliveryId>/redeliver",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
```

```
var res = await
        http.PostAsync("https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries/<deli
        veryId>/redeliver", null);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```
$ch =
        curl_init("https://api.snapkey.dk/public/v1/webhooks/<id>/deliveries/<deliveryI
        d>/redeliver");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```

At-least-once delivery

Delivery is **at-least-once** and unordered: dedupe on `data.id` together with `type`, and do not rely on deliveries arriving in the order the events happened.

Test locally

`POST /webhooks/{id}/ping` sends a test delivery — `{"message": "pong"}` under event type `ping` — to the subscription URL **synchronously** and returns the resulting delivery record, so you can see the exact HTTP status and error your endpoint produced. Use it to confirm your endpoint is reachable and your signature check works.

A ping is a single attempt: it is never retried, it does not clear a subscription's failure streak and it does not un-pause it.

curl

```
curl -X POST "https://api.snapkey.dk/public/v1/webhooks/<id>/ping" \
  -H "Authorization: Bearer $SNAPKEY_API_KEY"
```

Node

```
const res = await fetch("https://api.snapkey.dk/public/v1/webhooks/<id>/ping", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.SNAPKEY_API_KEY}`,
  },
});
const json = await res.json();
```

Python

```python
import os, requests

res = requests.post(
    "https://api.snapkey.dk/public/v1/webhooks/<id>/ping",
    headers={"Authorization": f"Bearer {os.environ['SNAPKEY_API_KEY']}"},
)
data = res.json()
```

C#

```csharp
using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
        Environment.GetEnvironmentVariable("SNAPKEY_API_KEY"));
var res = await http.PostAsync("https://api.snapkey.dk/public/v1/webhooks/<id>/ping",
        null);
var json = await res.Content.ReadAsStringAsync();
```

PHP

```php
$ch = curl_init("https://api.snapkey.dk/public/v1/webhooks/<id>/ping");
curl_setopt_array($ch, [
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        "Authorization: Bearer " . getenv("SNAPKEY_API_KEY"),
    ],
]);
$data = json_decode(curl_exec($ch), true);
```

The delivery your endpoint receives:

```json
{
  "id": "6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55",
  "type": "ping",
  "created_at": "2026-09-02T09:13:03Z",
  "data": {
    "message": "pong"
  }
}
```

The `url` must be `https://` and resolve to a public host, so a laptop needs a tunnel — ngrok, Cloudflare Tunnel or similar — rather than `localhost`.

What does not produce a webhook

`GET /events` exposes door events only: `access.granted`, `access.denied`, `door.closed` and `door.left_open`. Rows created outside the door-event pipeline — internal sync — and app telemetry are not events, so they are neither pushed to a subscription nor returned by `GET /events`.

`door.left_open` is only ever a real door-left-open report from the locking system: the lock did not report a close after being opened. It is never inferred by SnapKey.

Errors

The envelope

Every non-2xx response uses one envelope:

```
{
  "error": {
    "code": "validation_failed",
    "message": "The request could not be processed.",
    "details": [
      { "field": "phone", "code": "invalid", "message": "Phone must be in international
        format, e.g. +4520123456." }
    ]
  }
}
```

`details` is present only for `validation_failed`.

Status and code

Status	code	When
400	`invalid_request`	malformed query string or route parameters
401	`unauthenticated`	missing, invalid, expired or revoked API key
403	`insufficient_scope`	the key does not hold the scope this endpoint needs
403	`not_available`	reseller (platform-mode) account — not supported in v1
403	`forbidden`	the person is managed by SnapKey and cannot be modified
403	`lock_not_allowed`	the lock is not on this API key's allowlist
404	`not_found`	no such row — including rows outside the key's scope
405	`invalid_request`	that method is not supported on this path
409	`iloq_rejected`	the locking system refused the create, change or delete
409	`invitation_not_pending`	resend or cancel on an invitation that is activated, expired or cancelled
409	`idempotency_conflict`	this `Idempotency-Key` was already used for a different body
409	`idempotency_in_progress`	the first call with this `Idempotency-Key` is still running
409	`subscription_paused`	redeliver on a paused subscription
409	`lock_not_remote`	the lock is an iLOQ lock and cannot be opened over the network
409	`lock_offline`	the lock's device has not been heard from recently

Status	code	When
422	`validation_failed`	the body failed validation; see `details`
422	`entitlement_exceeded`	the account's plan limit for phone keys is reached — `POST /keys`, `POST /invitations/{id}/resend`
429	`rate_limited`	rate limit exceeded; see `Retry-After`
500	`server_error`	unexpected failure — safe to retry with backoff
502	`publish_failed`	the command could not be handed to the device
503	`not_available`	key revocation is not enabled on this server

Field-level details

Each entry in `details` names the offending `field` — e.g. `phone` or `security_groups[0]` — a `code` from the list below, and a `message` you can show or log.

- `invalid` — the value is wrong for that field, such as a phone number that is not in international format.
- `unknown_field` — the body carried a field the endpoint does not accept. Unknown fields are rejected rather than ignored, so typos surface at once.
- `reserved_name` — the `name` is one of SnapKey's reserved service names (`SnapKey`, `Admin`, `ServiceUser`), matched as the whole name or as the first word.
- `unknown_security_group` — a code in `security_groups` does not exist at the API key's location. The detail names the offending index.
- `sms_disabled` — SMS delivery was asked for on an account without SMS.
- `limit_reached` — the location already holds the maximum of 5 active webhook subscriptions.

What is safe to retry

- `server_error` (500) — an unexpected failure. Retry with backoff.
- `rate_limited` (429) — retry after the number of seconds in `Retry-After`.
- `idempotency_in_progress` (409) — the first call with this `Idempotency-Key` is still running. Retry after the 2 seconds in `Retry-After`.

Never retry these unchanged; they answer the same way every time:

- `validation_failed` (422) — fix the body first; `details` says what is wrong.
- `not_found` (404) — the row does not exist, or is outside the key's scope.
- `insufficient_scope` (403) — the key does not hold the scope this endpoint needs. Issue a key that does.

`iloq_rejected` (409) means the locking system refused the create, change or delete, and nothing was saved locally. Retry it only after checking the locking system — a blind retry meets the same refusal.

Use an `Idempotency-Key` on `POST /people`, `POST /keys` and `POST /webhooks` so a retry after a timeout cannot create a duplicate. Note that `5xx` responses and `409 iloq_rejected` are never stored, so a retry with the same key re-executes.

Changelog

v1.3.0 — 2026-09-19

Live status and remote unlock.

- `GET /locks/{id}`; `provider` and `online` on every lock
- `POST /locks/{id}/unlock`, `GET /locks/{id}/commands/{command_id}` — new scope `locks:control` with a per-key lock allowlist
- Door events carry `source` (`app`, `api`, `iloq`) and `api_key`; new reason `lock_not_allowed`
- New error codes: `lock_not_allowed`, `lock_not_remote`, `lock_offline`, `publish_failed`

v1.2.0 — 2026-09-19

Key lifecycle: read and update a key, and follow the invitation it was issued with.

- `GET /keys/{id}`, `PATCH /keys/{id}`; `invitation_id` on every key
- `GET /invitations`, `GET /invitations/{id}`, `POST /invitations/{id}/resend`, `DELETE /invitations/{id}`
- New error codes: `invitation_not_pending`, `entitlement_exceeded`; new validation detail `not_editable`

v1.1.0 — 2026-09-18

Subscription management, delivery history and seven new events.

- `GET /webhooks/{id}`, `PATCH /webhooks/{id}`
- `GET /webhooks/{id}/deliveries`, `POST /webhooks/{id}/deliveries/{delivery_id}/redeliver`
- Webhook events: `person.created`, `person.updated`, `person.deleted`, `key.issued`, `unlock.failed`, `lock.online`, `lock.offline`
- New error code: `subscription_paused`

v1.0.0 — 2026-09-04

First public release.

- `GET /locks`, `GET /security_groups`
- `GET /people`, `GET /people/{id}`, `POST /people`, `PUT /people/{id}`, `DELETE /people/{id}`
- `GET /keys`, `POST /keys`, `DELETE /keys/{id}`
- `GET /events`
- `GET /webhooks`, `POST /webhooks`, `DELETE /webhooks/{id}`, `POST /webhooks/{id}/ping`
- Webhook events: `access.granted`, `access.denied`, `door.closed`, `door.left_open`, `key.activated`, `key.revoked`, `ping`

SnapKey Public API (1.3.0)

Download OpenAPI specification: Download

Overview

The SnapKey Public API lets your own systems drive access control in SnapKey without anyone opening the dashboard. It carries two flows:

- **Your system → SnapKey.** Your HR or facility system creates and updates the **people** who need access, and issues **keys** to them. Issuing a key sends the person a setup link (SMS or e-mail); the key issued this way appears once they activate that link. `GET /keys` also lists keys that never had a setup link — physical cards and iLOQ S5 fobs managed elsewhere — so read `type` to tell them apart.
- **SnapKey → your system.** Read the access **events** from the doors (`GET /events`), or subscribe to **webhooks** and have SnapKey push each event to a URL you control.

Everything you can read or write is limited to the location your API key belongs to and the departments underneath it.

Authentication

Every request carries a bearer token:

```
Authorization: Bearer sk_live_9f3c…
```

API keys are created in the SnapKey dashboard under **Developer → API keys**. The full token is shown **once**, at creation — store it in your secret manager; SnapKey keeps only a hash and the `sk_live_9f3c` prefix. A key can be revoked at any time from the same screen; a revoked or expired key answers `401 unauthenticated`.

Every key carries an expiry date: **1 year** by default, up to **2 years** if the person creating it sets one further out. There is no non-expiring key — plan to rotate before `expires_at`, which `GET developer/api-keys` in the dashboard shows for every key.

A key carries an explicit list of scopes. A request to an endpoint whose scope the key does not hold answers `403 insufficient_scope`.

Scope	Grants
`catalog:read`	`GET /locks`, `GET /locks/{id}`, `GET /security_groups`
`people:read`	`GET /people`, `GET /people/{id}`
`people:write`	`POST`, `PUT`, `DELETE` on `/people` — and everything `people:read` grants
`keys:read`	`GET /keys`, `GET /keys/{id}`, `GET /invitations`, `GET /invitations/{id}`
`keys:write`	`POST /keys`, `PATCH`, `DELETE` on `/keys/{id}`, `POST /invitations/{id}/resend`, `DELETE /invitations/{id}` — and everything `keys:read` grants
`locks:control`	`POST /locks/{id}/unlock`, `GET /locks/{id}/commands/{command_id}` — only for the locks listed on the key
`events:read`	`GET /events`
`webhooks:manage`	all `/webhooks` endpoints

A read endpoint accepts either scope: `GET /people` is served for a key holding `people:read` **or** `people:write`.

Scope of a key

An API key belongs to exactly one location. Every list, lookup and write is limited to **that location and its descendants** — never the account root, never a sibling department. A row that exists elsewhere in the account is not "forbidden", it is simply `404 not_found`.

Reseller (platform-mode) accounts are not supported in v1: their keys answer `403 not_available` on every endpoint.

Rate limits

600 requests per minute per API key. A request that arrives without a bearer token is counted against a shared per-IP bucket of the same size instead.

Every response from an **authenticated** endpoint carries:

Header	Meaning
`X-RateLimit-Limit`	requests allowed per minute (600)
`X-RateLimit-Remaining`	requests left in the current window

Exceeding a limit answers `429 rate_limited` with a `Retry-After` header holding the number of seconds to wait. Two endpoints are limited harder, because each one makes SnapKey act on the outside world:

Endpoint	Limit
`POST /keys`	30 per minute, per API key — every call sends an SMS or e-mail
`POST /invitations/{id}/resend`	30 per minute, per API key — shares the POST /keys limit
`POST /webhooks/{id}/ping`	10 per minute, per API key
`POST /locks/{id}/unlock`	10 per minute, per API key, per lock

Requests that fail to authenticate are counted separately: **30 failed authentications per minute, per IP**. Once an IP has produced 30 failed authentications in a minute, every request from that IP is answered `429` until the window rolls — including requests with a valid key. `Retry-After` tells you how long. If you share an egress IP with other tenants or with misconfigured clients, expect this and back off.

Pagination

Every list endpoint returns a cursor page:

```
{ "data": [ … ], "next_cursor": "eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ
```

- `limit` — items per page, `1`–`200`, default `50`. Values outside the range are clamped.
- `cursor` — opaque; pass back the `next_cursor` of the previous page verbatim.

`next_cursor` is `null` on the last page. Rows are ordered by `id` ascending, so a page never reshuffles under you while you walk it.

Errors

Every non-2xx response uses one envelope:

```
{
  "error": {
    "code": "validation_failed",
    "message": "The request could not be processed.",
    "details": [
      { "field": "phone", "code": "invalid", "message": "Phone must be in internati
    ]
  }
}
```

`details` is present only for `validation_failed`.

Status	`code`	When
400	`invalid_request`	malformed query string or route parameters
401	`unauthenticated`	missing, invalid, expired or revoked API key
403	`insufficient_scope`	the key does not hold the scope this endpoint needs
403	`not_available`	reseller (platform-mode) account — not supported in v1
403	`forbidden`	the person is managed by SnapKey and cannot be modified
403	`lock_not_allowed`	the lock is not on this API key's allowlist

Status	`code`	When
404	`not_found`	no such row — including rows outside the key's scope
405	`invalid_request`	that method is not supported on this path
409	`iloq_rejected`	the locking system refused the create, change or delete
409	`invitation_not_pending`	resend or cancel on an invitation that is activated, expired or cancelled
409	`idempotency_conflict`	this `Idempotency-Key` was already used for a different body
409	`idempotency_in_progress`	the first call with this `Idempotency-Key` is still running
409	`subscription_paused`	redeliver on a paused subscription
409	`lock_not_remote`	the lock is an iLOQ lock and cannot be opened over the network
409	`lock_offline`	the lock's device has not been heard from recently
422	`validation_failed`	the body failed validation; see `details`
422	`entitlement_exceeded`	the account's plan limit for phone keys is reached — `POST /keys`, `POST /invitations/{id}/resend`
429	`rate_limited`	rate limit exceeded; see `Retry-After`
500	`server_error`	unexpected failure — safe to retry with backoff
502	`publish_failed`	the command could not be handed to the device
503	`not_available`	key revocation is not enabled on this server

Detail codes inside `details[].code`: `invalid`, `unknown_field`, `reserved_name`, `unknown_security_group`, `sms_disabled`, `limit_reached`, `not_editable`.

Idempotency

`POST /people`, `POST /keys` and `POST /webhooks` accept an optional `Idempotency-Key` request header — 1–64 characters of `A–Z a–z 0–9 _ -`. Send one per logical operation and reuse it when you retry after a timeout.

- The first request runs normally; its status and body are stored for **24 hours**, per API key.
- A retry with the same key **and the same body** returns the stored response verbatim, with `Idempotent-Replayed: true`. That is the answer to "did my retry create a duplicate?": no.
- The same key with a **different** body answers `409 idempotency_conflict`.
- A retry that arrives while the first call is still running answers `409 idempotency_in_progress` with `Retry-After: 2`.
- `5xx` responses and `409 iloq_rejected` are never stored — both are transient — so a retry with the same key re-executes.
- Responses larger than 64 KB, or that are not JSON, are not stored: a retry with the same key executes the request again (never a replay).
- A key that is still in progress after **60 seconds** is treated as abandoned (a killed worker, a dropped connection) and the retry re-executes rather than waiting out the 24 hours.

Without the header nothing changes: `POST /people` still de-duplicates on phone and e-mail, and `POST /keys` still sends every time.

Caching and retention

Every `public/v1` response carries `Cache-Control: no-store, private` — none of it may be cached by a proxy or a browser.

- Request logs (method, path, status, duration, IP — never bodies) are kept **90 days**.
- Webhook delivery payloads are kept **30 days**.
- Both are purged automatically. Deleting a person purges their delivery payloads immediately.

Timestamps

All timestamps are ISO 8601 in UTC with a `Z` suffix — `2026-09-02T08:15:30Z`. Plain dates use `YYYY-MM-DD`. Timestamps you send (`starts_at`, `expires_at`, `since`) are parsed as UTC unless they carry their own offset.

Webhook delivery

A subscription receives an HTTP `POST` for every matching event:

```
POST https://hooks.example.com/snapkey
Content-Type: application/json
User-Agent: SnapKey-Webhooks/1.0
X-SnapKey-Event: access.granted
X-SnapKey-Delivery: 6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55
X-SnapKey-Timestamp: 1788336930
X-SnapKey-Signature: t=<unix seconds>,v1=<hex HMAC-SHA256(secret, "{t}.{raw body}"

{ "id": "6f0a1d5c-…", "type": "access.granted", "created_at": "2026-09-02T08:15:30
```

Verify the signature like this:

1. Split `X-SnapKey-Signature` on `,` into `t=<unix seconds>` and `v1=<hex digest>`.
2. Reject the delivery if `t` is more than **300 seconds** away from your own clock — that is what stops a captured delivery from being replayed later.
3. Compute `HMAC-SHA256` over the string `"{t}.{raw body}"` — the raw bytes you received, not a re-serialised copy — keyed with your subscription secret.
4. Compare it with `v1` in **constant time** (`hash_equals`, `crypto.timingSafeEqual`, …).

```
[$t, $v1] = sscanf($header, 't=%d,v1=%s');
$valid = abs(time() - $t) <= 300
    && hash_equals(hash_hmac('sha256', $t.'.'.$rawBody, $secret), $v1);
```

Respond with any 2xx within 10 seconds. Anything else — a non-2xx status, a timeout, a connection failure — is a failed attempt. Redirects are not followed; any 3xx is a failed attempt. SnapKey retries after **1 m, 5 m, 30 m, 2 h, 6 h, 24 h**, then marks the delivery `exhausted` and stops.

A subscription that has been failing continuously for 24 hours is set to `paused`, and the user who created the API key (or the subscription) is e-mailed, when one is known. A paused subscription receives nothing; resume it from the dashboard and recover the gap with `GET /events`.

A subscription can also be managed entirely through the API: `PATCH /webhooks/{id}` pauses it (`status: paused`, stopping deliveries immediately) or resumes it (`status: active`, which also clears `failing_since`), and the same call changes `url`, `events` or `description`. `GET /webhooks/{id}/deliveries` lists past attempts — filterable by `status`, `event_type` and `since` — and `POST /webhooks/{id}/deliveries/{delivery_id}/redeliver` re-sends one as a fresh delivery with its own attempt ladder; the redelivery's `event_id` is the original delivery's id, so your duplicate guard recognises it as the same event. Redeliver answers `409 subscription_paused` while the subscription is paused — resume it first.

Delivery is **at-least-once** and unordered: dedupe on `data.id` together with `type`.

Locks

The doors in your account, and the security groups that open them.

List locks

Lists the locks in the API key's location and its departments.

Requires scope `catalog:read`.

AUTHORIZATIONS: › *ApiKey*

QUERY PARAMETERS

`location_id` integer

Example: `location_id=12`

Restrict to a single location inside the API key's scope.

`limit` integer

Default: `50`

Example: `limit=50`

Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.

`cursor` string

Example:

`cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`

Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

› **200** A page of locks.

› **401** Missing, invalid, expired or revoked API key.

› **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

› **422** The request failed validation.

› **429** Rate limit exceeded.

› **500** Unexpected failure. Safe to retry with backoff.

GET /locks

Request samples

curl

Copy

```
curl https://api.snapkey.dk/public/v1/locks?limit=50 \
  -H "Authorization: Bearer sk_live_9f3c…"
```

Response samples

200 | 401 | 403 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all

```
{
 - "data": [
    + { … }
   ],
   "next_cursor": null
}
```

Get a lock

One lock, same shape as the list. For a remote lock `online` reflects the device's last contact with SnapKey; poll it before an unlock, or subscribe to `lock.online` / `lock.offline`.

Requires scope `catalog:read`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	integer <int64> Example: `4172` The lock id.

Responses

> **200** The lock.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /locks/{id}

Response samples

200 401 403 404 429 500

Content type

application/json

Copy Expand all Collapse all


```
{
    "id": 4172,
    "name": "Main entrance",
    "serial_number": null,
    "place": "Ground floor, east",
    "provider": "teltonika",
    "online": true,
  - "security_groups": [
        "HQ-STAFF"
    ],
  - "location": {
        "id": 12,
        "name": "Headquarters"
    }
}
```


Open a lock

Opens a remote lock for its configured pulse. The API key must hold `locks:control` **and** list the lock on its allowlist (set when the key is created in the dashboard) — a lock the key may not open answers `403 lock_not_allowed` and is recorded as a refused access.

The call answers **202 Accepted** with a command: the device has been told, not yet heard from. Poll `GET /locks/{id}/commands/{command_id}`, or let the outcome reach you as `access.granted` or `unlock.failed` on a webhook. A second call while a command is still in flight returns that same command instead of pulsing the door again.

Rate limited to **10 requests per minute, per API key, per lock**. Requires scope `locks:control`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	integer <int64>	Example: `4172`

The lock id.

Responses

- **202** The command was sent to the device.
- **401** Missing, invalid, expired or revoked API key.
- **403** The key lacks the scope, or the lock is not on its allowlist.
- **404** No such row — including rows that exist outside the API key's scope.
- **409** The lock cannot be opened right now. Nothing was sent.
- **429** Rate limit exceeded.
- **500** Unexpected failure. Safe to retry with backoff.
- **502** The command could not be handed to the device. It is recorded as `unlock.failed`.

POST /locks/{id}/unlock

Request samples

curl

Copy

```
curl -X POST https://api.snapkey.dk/public/v1/locks/4172/unlock \
  -H "Authorization: Bearer sk_live_9f3c…"
```

Response samples

202 | 401 | 403 | 404 | 409 | 429 | 500 | 502

Content type

application/json

Copy


```
{
    "id": "0d3f6c2a-7b1e-4f0a-9c8d-2e5b7a1f4c33",
    "lock_id": 4172,
    "status": "published",
    "requested_at": "2026-09-19T09:40:11Z",
    "published_at": "2026-09-19T09:40:11Z",
    "confirmed_at": null,
    "timeout_seconds": 10,
    "error": null
}
```


Get an unlock command

The state of a command returned by `POST /locks/{id}/unlock`. `status` moves from `published` to `confirmed` when the device answers, or to `timed_out` after `timeout_seconds` without an answer; `failed` means the device could not be reached at all.

Requires scope `locks:control` and the lock on the key's allowlist.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

- `id` (required) — integer <int64>
  Example: `4172`
  The lock id.
- `commandId` (required) — string <uuid>
  Example: `0d3f6c2a-7b1e-4f0a-9c8d-2e5b7a1f4c33`

Responses

> **200** The command.

> **401** Missing, invalid, expired or revoked API key.

403
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

404 No such row — including rows that exist outside the API key's scope.

429 Rate limit exceeded.

500 Unexpected failure. Safe to retry with backoff.

GET `/locks/{id}/commands/{commandId}`

Response samples

200 | 401 | 403 | 404 | 429 | 500

Content type
application/json

Copy


```
{
    "id": "0d3f6c2a-7b1e-4f0a-9c8d-2e5b7a1f4c33",
    "lock_id": 4172,
    "status": "confirmed",
    "requested_at": "2026-09-19T09:40:11Z",
    "published_at": "2026-09-19T09:40:11Z",
    "confirmed_at": "2026-09-19T09:40:12Z",
    "timeout_seconds": 10,
    "error": null
}
```


Security groups

The access profiles a key can be issued against.

List security groups

Lists the security groups (access profiles) in the API key's location and its departments. Their `code` values are what you pass to `POST /keys`.

Requires scope `catalog:read`.

AUTHORIZATIONS: > *ApiKey*

QUERY PARAMETERS

- `is_default` — boolean
  Example: `is_default=true`
  Return only the default (`true`) or only the non-default (`false`) groups.
- `limit` — integer
  Default: `50`
  Example: `limit=50`
  Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.
- `cursor` — string
  Example: `cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`
  Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

> **200** A page of security groups.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **422** The request failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /security_groups

Response samples

200 | 401 | 403 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
 - "data": [
     + { … }
   ],
   "next_cursor": null
}
```


People

The employees, contractors and guests who hold keys.

List people

Lists the people in the API key's location and its departments. `email` matches case-insensitively; `phone` is normalised before matching, so `+45 20 12 34 56` and `+4520123456` find the same person.

Requires scope `people:read` or `people:write`.

AUTHORIZATIONS: > *ApiKey*

QUERY PARAMETERS

`email`
string
Example: `email=mette.sorensen@example.com`
Exact, case-insensitive e-mail match.

`phone`
string
Example: `phone=+4520123456`
Phone number in any format; normalised before matching.

`limit`
integer
Default: `50`
Example: `limit=50`
Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.

`cursor`
string
Example:
`cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`
Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

> **200** A page of people.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **422** The request failed validation.

> **429** Rate limit exceeded.

› **500** Unexpected failure. Safe to retry with backoff.

GET /people

Response samples

200 | 401 | 403 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
 - "data": [
     + { … }
   ],
   "next_cursor": null
}
```


Create a person

Creates a person and, when the account syncs with iLOQ, creates the matching person there.

Idempotent on phone and e-mail. If a person in the API key's scope already has the same normalised phone number or the same e-mail address, that person is returned with **200 OK** and nothing is changed. A genuinely new person answers **201 Created**. Send the request again after a timeout without fear of duplicates.

Unknown fields are rejected with `422 validation_failed` so typos surface at once. A `name` reserved for SnapKey's own service accounts (`SnapKey`, `Admin`, `ServiceUser`, matched whole or as the first word) is rejected the same way with detail code `reserved_name`.

Requires scope `people:write`.

AUTHORIZATIONS: › *ApiKey*

HEADER PARAMETERS

Idempotency-Key

string `^[A-Za-z0-9_-]{1,64}$`

Example: `order-4711`

Retry-safety token for this operation, 1–64 characters of `A-Za-z0-9_-`, unique per logical request. A repeat with the same key and body replays the stored response (`Idempotent-Replayed: true`); a repeat with a different body answers `409 idempotency_conflict`. Kept 24 hours, scoped to the API key. Responses larger than 64 KB, or that are not JSON, are not stored: a retry with the same key executes the request again (never a replay).

REQUEST BODY SCHEMA: application/json

required

name
required

string <= 101 characters

First and last name, separated by whitespace.

email
required

string <email> <= 255 characters

phone
required

string `^\+?[0-9 ]{6,20}$`

International format, e.g. `+4520123456`.

company_name

string or null <= 100 characters

title

string or null <= 100 characters

language

string or null

Enum: `"da"` `"en"` `null`

Language of the setup link and reminders. Defaults to the location's language.

location_id

integer or null

Which location to place the person in. Must be inside the API key's scope. Defaults to the API key's own location.

Responses

> **200** A person with this phone number or e-mail already existed; it is returned unchanged.

> **201** The person was created.

> **401** Missing, invalid, expired or revoked API key.

403
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

409
The locking system refused to create this person (`iloq_rejected`), this Idempotency-Key was already used for a different request (`idempotency_conflict`), or the first call with it is still running (`idempotency_in_progress`).

422 The body failed validation.

429 Rate limit exceeded.

500 Unexpected failure. Safe to retry with backoff.

POST `/people`

Request samples

Payload | curl

Content type
application/json

Copy

```
{
    "name": "Mette Sørensen",
    "email": "mette.sorensen@example.com",
    "phone": "+4520123456",
    "company_name": "Tidevand Energi",
    "title": "Facility Manager",
    "language": "da",
    "location_id": 12
}
```

Response samples

200 | 201 | 401 | 403 | 409 | 422 | 429 | 500

Content type

application/json

Copy Expand all Collapse all


```
{
   "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
   "name": "Mette Sørensen",
   "email": "mette.sorensen@example.com",
   "phone": "+4520123456",
   "company_name": "Tidevand Energi",
   "title": "Facility Manager",
   "language": "da",
 - "location": {
      "id": 12,
      "name": "Headquarters"
   }
}
```


Get a person

Returns one person, including the `keys` they currently hold inside the API key's scope.

Requires scope `people:read` or `people:write`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	string <uuid> Example: `9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40` The person's UUID.

Responses

> **200** The person, with their keys.

> **401** Missing, invalid, expired or revoked API key.

403
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

404 No such row — including rows that exist outside the API key's scope.

429 Rate limit exceeded.

500 Unexpected failure. Safe to retry with backoff.

GET /people/{id}

Response samples

200 | 401 | 403 | 404 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
   "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
   "name": "Mette Sørensen",
   "email": "mette.sorensen@example.com",
   "phone": "+4520123456",
   "company_name": "Tidevand Energi",
   "title": "Facility Manager",
   "language": "da",
 - "location": {
      "id": 12,
      "name": "Headquarters"
   },
 - "keys": [
    + { … }
   ]
}
```

Update a person

Updates a person. Every field is optional; only the fields you send are changed. Unknown fields are rejected with `422 validation_failed`, as is a `name` that is one of SnapKey's reserved service names (`SnapKey`, `Admin`, `ServiceUser`, as the whole name or as the first word) — detail code `reserved_name`.

The change is written to the locking system first — if iLOQ refuses it, nothing is saved locally and the call answers `409 iloq_rejected`.

SnapKey's own service people cannot be modified through the public API and answer `403 forbidden`.

Requires scope `people:write`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

- `id` (required) — string <uuid>
  Example: `9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40`
  The person's UUID.

REQUEST BODY SCHEMA: application/json (required)

- `name` — string <= 101 characters
- `email` — string <email> <= 255 characters
- `phone` — string `^\+?[0-9 ]{6,20}$`
- `company_name` — string or null <= 100 characters
- `title` — string or null <= 100 characters
- `language` — string or null
  Enum: `"da"` `"en"` `null`
- `location_id` — integer
  Must be inside the API key's scope.

Responses

200 The updated person.

401 Missing, invalid, expired or revoked API key.

403
The key lacks the `people:write` scope (`insufficient_scope`), the account is a reseller account (`not_available`), or this person is managed by SnapKey (`forbidden`).

404 No such row — including rows that exist outside the API key's scope.

409 The locking system refused this change. Nothing was saved.

422 The request failed validation.

429 Rate limit exceeded.

500 Unexpected failure. Safe to retry with backoff.

PUT `/people/{id}`

Request samples

Payload

Content type
application/json

Copy

```
{
   "title": "Head of Facilities",
   "phone": "+4520998877"
}
```

Response samples

200 | 401 | 403 | 404 | 409 | 422 | 429 | 500

Content type

application/json

Copy Expand all Collapse all


```
{
   "id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
   "name": "Mette Sørensen",
   "email": "mette.sorensen@example.com",
   "phone": "+4520123456",
   "company_name": "Tidevand Energi",
   "title": "Facility Manager",
   "language": "da",
 - "location": {
      "id": 12,
      "name": "Headquarters"
   }
}
```


Delete a person

Deletes a person. When the account syncs with iLOQ the person is deleted there first; if the locking system refuses, nothing is deleted and the call answers `409 iloq_rejected`.

SnapKey's own service people cannot be deleted through the public API and answer `403 forbidden`.

Requires scope `people:write`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	string <uuid> Example: `9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40` The person's UUID.

Responses

— **204** The person was deleted. No body.

› **401** Missing, invalid, expired or revoked API key.

› **403**
The key lacks the `people:write` scope (`insufficient_scope`), the account is a reseller account (`not_available`), or this person is managed by SnapKey (`forbidden`).

› **404** No such row — including rows that exist outside the API key's scope.

› **409** The locking system refused to delete this person. Nothing was deleted.

› **429** Rate limit exceeded.

› **500** Unexpected failure. Safe to retry with backoff.

DELETE /people/{id}

Response samples

401 | 403 | 404 | 409 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
 - "error": {
      "code": "unauthenticated",
      "message": "Missing, invalid, expired or revoked API key."
   }
}
```

Keys

Access grants, and the invitations that create them.

List keys

Lists **every** access grant in the API key's location and its departments — not only the ones issued through this API. A key issued with `POST /keys` appears here once the person activates their setup link; until then the invitation is all there is. Physical cards and iLOQ S5 fobs, which are managed outside this API and never had a setup link, are listed too. Read `type` to tell them apart.

Requires scope `keys:read` or `keys:write`.

AUTHORIZATIONS: › *ApiKey*

QUERY PARAMETERS

`person_id`
string <uuid>
Example: `person_id=9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40`
Only the keys held by this person.

`state`
string
Example: `state=handed_over`
Only keys in this state — `planning`, `sent`, `handed_over`, `returned` or `other`. An unrecognised value is not rejected; it simply matches nothing and returns an empty page.

`security_group`
string
Example: `security_group=HQ-STAFF`
Only keys that carry this security group code.

`limit`
integer
Default: `50`
Example: `limit=50`
Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.

cursor

string

Example:

`cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`

Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

> **200** A page of keys.

> **401** Missing, invalid, expired or revoked API key.

> **403**

The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **422** The request failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /keys

Response samples

200 | 401 | 403 | 422 | 429 | 500

Content type

application/json

Copy Expand all Collapse all


```
{
  - "data": [
      + { … }
    ],
    "next_cursor": null
```

```
}
```

Issue a key (invitation)

Issues a key to a person. SnapKey creates an **invitation** and sends the person a setup link over SMS, e-mail or both; the call answers **202 Accepted** with the invitation, not a key. Follow it with `GET /invitations/{id}`, resend it with `POST /invitations/{id}/resend`, or recall it with `DELETE /invitations/{id}`. The key itself appears in `GET /keys` with state `handed_over` once the person activates the link, and fires the `key.activated` webhook at that moment.

`security_groups` must be codes that exist at the API key's location — an unknown code answers `422` with detail code `unknown_security_group`, naming the offending index. Asking for SMS delivery on an account without SMS answers `422` with detail code `sms_disabled`. Answers `422 entitlement_exceeded` when the account's phone-key allowance is used up.

Rate limited to **30 requests per minute, per API key**, counted before validation and scope checks — a rejected request still consumes one. `X-RateLimit-Limit` on this route shows the limiter closest to exhaustion on that route, so they can show `30` rather than the account-wide 600.

Requires scope `keys:write`.

AUTHORIZATIONS: > *ApiKey*

HEADER PARAMETERS

- `Idempotency-Key` — string `^[A-Za-z0-9_-]{1,64}$`
  Example: `order-4711`
  Retry-safety token for this operation, 1–64 characters of `A-Za-z0-9_-`, unique per logical request. A repeat with the same key and body replays the stored response (`Idempotent-Replayed: true`); a repeat with a different body answers `409 idempotency_conflict`. Kept 24 hours, scoped to the API key. Responses larger than 64 KB, or that are not JSON, are not stored: a retry with the same key executes the request again (never a replay).

REQUEST BODY SCHEMA: application/json
required

- `person_id` (required) — string <uuid>
  The person to issue the key to. Must be inside the API key's scope.
- `security_groups` (required) — Array of strings non-empty [items <= 64 characters]

Security group codes from `GET /security_groups`. All must exist at the API key's location.

`name`

string or null `<= 50 characters`

Label the person sees. Defaults to the title of the first security group.

`starts_at`

string or null <date-time>

When the key becomes valid. Omit for immediately.

`expires_at`

string or null <date-time>

When the key stops working. Must be after `starts_at`. Omit for never.

`channel`

string or null

Default: `"sms"`

Enum: `"sms"` `"email"` `"both"` `null`

How the setup link is delivered. `sms` and `both` require SMS to be enabled for the account.

Responses

> **202** The invitation was created and the setup link dispatched.

> **401** Missing, invalid, expired or revoked API key.

> **403**

The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such person in the API key's scope.

> **409**

This `Idempotency-Key` was already used for a different request body (`idempotency_conflict`), or the first call with it is still running (`idempotency_in_progress`).

> **422** The body failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

POST /keys

Request samples

Payload | curl

Content type
application/json

Copy Expand all Collapse all


```
{
   "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
 - "security_groups": [
      "HQ-STAFF"
   ],
   "name": "Headquarters staff",
   "starts_at": "2026-09-01T00:00:00Z",
   "expires_at": "2027-09-01T00:00:00Z",
   "channel": "sms"
}
```


Response samples

202 | 401 | 403 | 404 | 409 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
   "invitation_id": 5521,
   "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
 - "security_groups": [
      "HQ-STAFF"
   ],
   "state": "sent",
 - "sent": {
      "email": false,
      "sms": true
   }
```

```
}
```

Get a key

One key, same shape as the list. Revoked keys are gone from here as well as from the list.

Requires scope `keys:read` or `keys:write`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	integer <int64> Example: `88213` The key id.

Responses

> **200** The key.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /keys/{id}

Response samples

200 | 401 | 403 | 404 | 429 | 500

Content type

application/json

Copy Expand all Collapse all


```
{
    "id": 88213,
    "name": "Headquarters staff",
    "type": "digital",
    "state": "handed_over",
    "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "invitation_id": 5521,
  - "security_groups": [
        "HQ-STAFF"
    ],
    "starts_at": "2026-09-01T00:00:00Z",
    "expires_at": null
}
```


Update a key

Changes a key's name, security groups or validity window. Send only the fields you want to change; at least one is required.

Any change other than a pure rename puts the key back into state `sent` until the locking system confirms it, at which point it returns to `handed_over` and fires `key.activated` again. An iLOQ S5 fob is managed in iLOQ Manager: only `name` can be changed, every other field answers `422` with detail code `not_editable`.

When the account syncs with iLOQ and the key is already linked, the change is written to iLOQ **before** it is saved; a refusal answers `409 iloq_rejected` and nothing changes. Editing a key through the API replaces any time limits set directly in iLOQ Manager. Sending the same security groups in a different order is not a change.

Requires scope `keys:write`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

- `id` (required)
  integer <int64>
  Example: `88213`
  The key id.

REQUEST BODY SCHEMA: application/json
required

- `name`
  string or null <= 50 characters
- `security_groups`
  Array of strings non-empty [items <= 64 characters]
  Replaces the key's security groups. Codes must exist at the API key's location.
- `starts_at`
  string or null <date-time>
  `null` means valid immediately.
- `expires_at`
  string or null <date-time>
  Must be after `starts_at`. `null` means never.

Responses

> **200** The updated key.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **409** The locking system refused the change. Nothing was saved.

> **422** The body failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

PATCH /keys/{id}

Request samples

Payload | curl

Content type
application/json

Copy Expand all Collapse all


```
{
 - "security_groups": [
      "HQ-STAFF",
      "HQ-PARKING"
   ],
   "expires_at": "2027-09-01T00:00:00Z"
}
```


Response samples

200 | 401 | 403 | 404 | 409 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all

```
{
    "id": 88213,
    "name": "Headquarters staff",
    "type": "digital",
    "state": "sent",
    "person_id": "9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40",
    "invitation_id": 5521,
  - "security_groups": [
        "HQ-STAFF",
        "HQ-PARKING"
    ],
    "starts_at": "2026-09-01T00:00:00Z",
    "expires_at": "2027-09-01T00:00:00Z"
}
```


Revoke a key

Revokes a key. The key expires immediately, is removed from SnapKey and is queued for removal in the locking system; the `key.revoked` webhook fires at once. Revocation in the locking system finishes asynchronously.

Requires scope `keys:write`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	integer <int64> Example: `88213` The key id.

Responses

— **204** The key was revoked. No body.

> **401** Missing, invalid, expired or revoked API key.

403
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

404 No such row — including rows that exist outside the API key's scope.

429 Rate limit exceeded.

500 Unexpected failure. Safe to retry with backoff.

503 Key revocation is not enabled on this server. Nothing was changed.

DELETE `/keys/{id}`

Response samples

401 | 403 | 404 | 429 | 500 | 503

Content type
application/json

Copy Expand all Collapse all


```
{
  - "error": {
        "code": "unauthenticated",
        "message": "Missing, invalid, expired or revoked API key."
    }
}
```


Events

The access log — who opened which door, when, and what was refused.

List access events

The access log for the locks in the API key's location and its departments, oldest first (`id` ascending — the order SnapKey received them in). This is the endpoint to poll, and the endpoint to replay from after a webhook gap.

Only door events are exposed: `access.granted`, `access.denied`, `door.closed` and `door.left_open`. Internal sync and app telemetry rows are never returned.

Door events only. `unlock.failed`, key, person and lock presence events are webhook-only.

Requires scope `events:read`.

AUTHORIZATIONS: > *ApiKey*

QUERY PARAMETERS

- `since` — string <date-time>
  Example: `since=2026-09-01T00:00:00Z`
  Only events at or after this instant. ISO 8601; parsed as UTC when no offset is given.
- `lock_id` — integer <int64>
  Example: `lock_id=4172`
  Only events from this lock.
- `person_id` — string <uuid>
  Example: `person_id=9c1f2a84-3b7e-4d21-9f60-5a2c8d7e1b40`
  Only events caused by this person.
- `type` — string
  Example: `type=access.granted,access.denied`
  Comma-separated list of event types. An unrecognised list matches nothing.
- `limit` — integer
  Default: `50`
  Example: `limit=50`
  Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.
- `cursor` — string

Example:

`cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`

Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

> **200** A page of events.

> **401** Missing, invalid, expired or revoked API key.

> **403**

The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **422** The request failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /events

Request samples

curl

Copy

```
curl -G https://api.snapkey.dk/public/v1/events \
  -H "Authorization: Bearer sk_live_9f3c…" \
  --data-urlencode "since=2026-09-01T00:00:00Z" \
  --data-urlencode "type=access.granted,access.denied" \
  --data-urlencode "limit=200"
```

Response samples

200 401 403 422 429 500

Content type
application/json

Copy Expand all Collapse all


```
{
  - "data": [
      + { … },
      + { … }
    ],
    "next_cursor": "eyJpZCI6OTkxMDQzMywiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ"
}
```


Webhooks

Push subscriptions that deliver events to a URL you control.

List webhook subscriptions

Lists the webhook subscriptions of the API key's own location. The `secret` is never returned here — it is shown once, at creation.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

QUERY PARAMETERS

`limit` integer
Default: `50`
Example: `limit=50`
Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.

cursor

string

Example:

`cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`

Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

> **200** A page of subscriptions.

> **401** Missing, invalid, expired or revoked API key.

> **403**

The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /webhooks

Response samples

200 | 401 | 403 | 429 | 500

Content type

application/json

Copy Expand all Collapse all


```
{
  - "data": [
      + { … }
    ],
    "next_cursor": null
}
```

Create a webhook subscription

Subscribes a URL to one or more event types. Pass `"*"` to receive every type.

The response carries the signing `secret` **once** — store it now; it is never returned again. The secret is always generated by SnapKey; a `secret` field in the request is rejected as an unknown field.

The `url` must be `https://` and resolve to a public host; loopback, private-range and `.local` / `.internal` hosts are refused. The API key's own location may hold at most 5 active subscriptions — the sixth answers `422` with detail code `limit_reached`. The cap is per location, so departments under it have their own allowance.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

HEADER PARAMETERS

- `Idempotency-Key`
  string `^[A-Za-z0-9_-]{1,64}$`
  Example: `order-4711`
  Retry-safety token for this operation, 1–64 characters of `A-Za-z0-9_-`, unique per logical request. A repeat with the same key and body replays the stored response (`Idempotent-Replayed: true`); a repeat with a different body answers `409 idempotency_conflict`. Kept 24 hours, scoped to the API key. Responses larger than 64 KB, or that are not JSON, are not stored: a retry with the same key executes the request again (never a replay).

REQUEST BODY SCHEMA: application/json
required

- `url`
  required
  string <uri> <= 2048 characters `^https://`
  Must be `https://` and resolve to a public host.
- `events`
  required
  Array of strings (WebhookEventName) non-empty
  Items Enum: `"access.granted"` `"access.denied"` `"door.closed"` `"door.left_open"` `"key.activated"` `"key.revoked"` `"unlock.failed"` `"key.issued"` `"person.created"` `"person.updated"` `"person.deleted"` `"lock.online"` `"lock.offline"` `"*"`
- `description`
  string or null <= 255 characters

Responses

> **201** The subscription was created. `secret` appears in this response only.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **409**
This `Idempotency-Key` was already used for a different request body (`idempotency_conflict`), or the first call with it is still running (`idempotency_in_progress`).

> **422** The body failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

POST `/webhooks`

Request samples

Payload | **curl**

Content type
application/json

Copy Expand all Collapse all

```
{
   "url": "https://hooks.example.com/snapkey",
 - "events": [
      "access.granted",
      "access.denied"
   ],
   "description": "Tidevand Energi facility dashboard"
}
```


Response samples

201 | 401 | 403 | 409 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
   "id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
   "url": "https://hooks.example.com/snapkey",
 - "events": [
      "access.granted",
      "access.denied"
   ],
   "description": "Tidevand Energi facility dashboard",
   "status": "active",
   "failing_since": null,
   "last_delivery_at": null,
   "created_at": "2026-09-02T09:12:00Z",
   "secret": "whsec_3f8c1d90a47b4e2fa6c5029d81be7a34"
}
```


Get a webhook subscription

One subscription, same shape as the list. The `secret` is never returned here.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id
required

string <uuid>
Example: `4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1`
The subscription's UUID.

Responses

200 The subscription.

401 Missing, invalid, expired or revoked API key.

403
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

404 No such row — including rows that exist outside the API key's scope.

429 Rate limit exceeded.

500 Unexpected failure. Safe to retry with backoff.

GET /webhooks/{id}

Response samples

200 | 401 | 403 | 404 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
    "id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
    "url": "https://hooks.example.com/snapkey",
```

```
  - "events": [
        "access.granted",
        "access.denied"
    ],
    "description": "Tidevand Energi facility dashboard",
    "status": "active",
    "failing_since": null,
    "last_delivery_at": "2026-09-02T08:15:31Z",
    "created_at": "2026-08-20T09:00:00Z"
}
```

Update a webhook subscription

Changes any of `url`, `events`, `description`, `status`. At least one field is required.

Setting `status` to `paused` stops deliveries immediately (in-flight retries fail with "Subscription is paused."). Setting it to `active` on a paused subscription resumes it and clears `failing_since` — the same effect as the dashboard's Resume button; on an already active subscription it is a no-op. `url` and `events` are validated exactly as on create; changing `url` does not rotate the secret.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: › *ApiKey*

PATH PARAMETERS

- `id` — required
  string <uuid>
  Example: `4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1`
  The subscription's UUID.

REQUEST BODY SCHEMA: application/json
required

- `url`
  string <uri> <= 2048 characters `^https://`
  Must be `https://` and resolve to a public host. Does not rotate the secret.
- `events`
  Array of strings (WebhookEventName) non-empty
  Items Enum: `"access.granted"` `"access.denied"` `"door.closed"` `"door.left_open"` `"key.activated"` `"key.revoked"` `"unlock.failed"` `"key.issued"` `"person.created"`

	`"person.updated"` `"person.deleted"` `"lock.online"` `"lock.offline"` `"*"`
`description`	string or null <= 255 characters
`status`	string (WebhookStatus) Enum: `"active"` `"paused"` `active` — delivering. `paused` — failing for 24 hours straight; SnapKey stopped sending and e-mailed the user who created it, when one is known. Resume it from the dashboard, then replay the gap with `GET /events`.

Responses

> **200** The updated subscription.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **422** The body failed validation.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

PATCH `/webhooks/{id}`

Request samples

Payload

Content type

application/json

Copy

```
{
   "status": "active"
}
```

Response samples

200 | 401 | 403 | 404 | 422 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
   "id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
   "url": "https://hooks.example.com/snapkey",
 - "events": [
      "access.granted",
      "access.denied"
   ],
   "description": "Tidevand Energi facility dashboard",
   "status": "active",
   "failing_since": null,
   "last_delivery_at": "2026-09-02T08:15:31Z",
   "created_at": "2026-08-20T09:00:00Z"
}
```


Delete a webhook subscription

Deletes a subscription. Deliveries stop immediately.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	string <uuid> Example: `4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1`

The subscription's UUID.

Responses

— **204** The subscription was deleted. No body.

› **401** Missing, invalid, expired or revoked API key.

› **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

› **404** No such row — including rows that exist outside the API key's scope.

› **429** Rate limit exceeded.

› **500** Unexpected failure. Safe to retry with backoff.

DELETE /webhooks/{id}

Response samples

401 | 403 | 404 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
 - "error": {
      "code": "unauthenticated",
      "message": "Missing, invalid, expired or revoked API key."
   }
}
```

Ping a webhook subscription

Sends a test delivery — `{"message": "pong"}` under event type `ping` — to the subscription URL **synchronously** and returns the resulting delivery record, so you can see the exact HTTP status and error your endpoint produced.

A ping is a single attempt: it is never retried, it does not clear a subscription's failure streak and it does not un-pause it. Pings are limited to **10 per minute**.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	string <uuid> Example: `4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1` The subscription's UUID.

Responses

> **200** The delivery record for the ping.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **429** More than 10 pings in a minute. Retry after the number of seconds in `Retry-After`.

> **500** Unexpected failure. Safe to retry with backoff.

POST /webhooks/{id}/ping

Response samples

200 | 401 | 403 | 404 | 429 | 500

Content type

application/json

Copy Expand all Collapse all


```
{
   "id": "6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55",
   "subscription_id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
   "event_type": "ping",
   "event_id": null,
   "status": "delivered",
   "attempt": 1,
   "response_status": 200,
   "last_error": null,
   "next_attempt_at": null,
   "delivered_at": "2026-09-02T09:13:04Z",
   "created_at": "2026-09-02T09:13:03Z",
 - "payload": {
      "message": "pong"
   }
}
```


List a subscription's deliveries

A cursor page of delivery attempts against this subscription, newest first. Retention is **30 days**.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	string <uuid> Example: `4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1` The subscription's UUID.

QUERY PARAMETERS

`status`

string (DeliveryStatus)

Enum: `"pending"` `"retrying"` `"delivered"` `"exhausted"` `"failed"`

Only deliveries in this status.

`event_type`

string

Example: `event_type=access.granted`

Only deliveries of this event type. Any name from `WebhookEventName`, plus `ping`, which is a delivery type rather than a subscribable event.

`since`

string <date-time>

Example: `since=2026-09-01T00:00:00Z`

Only deliveries created at or after this instant. ISO 8601; parsed as UTC when no offset is given.

`limit`

integer

Default: `50`

Example: `limit=50`

Items per page. Values below 1 or above 200 are clamped into that range rather than rejected; a non-integer is rejected with `422 validation_failed` — except on `GET /webhooks`, which does not validate `limit` and simply clamps whatever it can cast.

`cursor`

string

Example: `cursor=eyJpZCI6NDE3MiwiX3BvaW50c1RvTmV4dEl0ZW1zIjp0cnVlfQ`

Opaque page cursor — the `next_cursor` of the previous page, passed back verbatim.

Responses

> **200** A page of deliveries.

> **400** An invalid `status` or `since` value.

> **401** Missing, invalid, expired or revoked API key.

> **403**

The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **429** Rate limit exceeded.

> **500** Unexpected failure. Safe to retry with backoff.

GET /webhooks/{id}/deliveries

Response samples

200 | 400 | 401 | 403 | 404 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
 - "data": [
     + { … }
   ],
   "next_cursor": null
}
```


Redeliver a delivery

Sends the same `event_type` and payload again as a brand-new delivery, `202` at the default attempt (0), so it follows the full retry ladder from the start. The original delivery row is untouched; the new one's `event_id` is the original delivery's id, so your duplicate guard treats it as the same event. Works for any original status, including `delivered` and pings.

Only on an `active` subscription — otherwise `409 subscription_paused`. Limited to **30 per minute, per API key**.

Requires scope `webhooks:manage`.

AUTHORIZATIONS: > *ApiKey*

PATH PARAMETERS

id required	string <uuid> Example: `4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1` The subscription's UUID.
deliveryId required	string <uuid> The delivery to redeliver.

Responses

> **202** The new delivery, queued.

> **401** Missing, invalid, expired or revoked API key.

> **403**
The key does not hold the scope this endpoint needs (`insufficient_scope`), or the account is a reseller (platform-mode) account, which v1 does not support (`not_available`).

> **404** No such row — including rows that exist outside the API key's scope.

> **409** The subscription is paused.

> **429** More than 30 redeliveries in a minute. Retry after the number of seconds in `Retry-After`.

> **500** Unexpected failure. Safe to retry with backoff.

POST /webhooks/{id}/deliveries/{deliveryId}/redeliver

Response samples

202 | 401 | 403 | 404 | 409 | 429 | 500

Content type
application/json

Copy Expand all Collapse all


```
{
   "id": "9b3e5f71-1a4c-4e3a-8b0d-6f2c7a91de44",
```

```
    "subscription_id": "4d2a91c6-8f35-4b0e-9a17-63d8c0f5e2b1",
    "event_type": "access.granted",
    "event_id": "6f0a1d5c-2b47-4a19-8f31-7c9de2b04a55",
    "status": "pending",
    "attempt": 1,
    "response_status": null,
    "last_error": null,
    "next_attempt_at": "2026-09-02T09:20:00Z",
    "delivered_at": null,
    "created_at": "2026-09-02T09:20:00Z",
  - "payload": {
        "id": 9910427,
        "type": "access.granted",
        "occurred_at": "2026-09-02T08:15:30Z"
    }
}
```