REST API reference

This reference describes the client-facing operations exposed by the current SafeDNS Shield REST API.

Using the API

Base URL

SafeDNS Shield does not have a universal API hostname. The management API address is assigned during deployment.

```
<base-url> = http[s]://<shield-management-host>:<management-api-port>
```

The default management API port is `8080`, but it may be changed for a particular deployment.

Authentication

All operations in this reference require HTTP bearer authentication.

```
Authorization: Bearer <token>
Content-Type: application/json
```

Example:

```
curl \
  --header "Authorization: Bearer <token>" \
  --header "Content-Type: application/json" \
  "<base-url>/profiles/"
```

Access may additionally be restricted by source IP address according to the deployment configuration.

General conventions

Immediate application

Successfully completed changes take effect immediately. No service restart, reload, or separate activation operation is required.

Success status codes

Status	Typical use
`200 OK`	Successful retrieval or update
`201 Created`	Successful resource creation
`204 No Content`	Successful deletion or initialization; no response body

Validation errors

Invalid request data may return `422 Unprocessable Entity`.

Example response:

```
{
  "detail": [
    {
      "loc": [
        "body",
        "field_name"
      ],
      "msg": "Validation error message",
      "type": "validation_error"
    }
  ]
}
```

The current OpenAPI specification formally describes validation errors but does not provide complete response schemas for every possible authentication, authorization, conflict, or not-found condition.

IPv4 representations

IPv4 addresses appear in two forms:

- **Dotted-decimal string**, for example `"100.100.100.101"`, in request and response bodies.
- **Unsigned 32-bit integer**, for example `1684300901`, in individual IPv4 and NAPT resource paths.

Python conversion example:

```
from ipaddress import IPv4Address


int_ip = int(IPv4Address("100.100.100.101"))
```

```
ip = str(IPv4Address(1684300901))
```

Category IDs

Filtering profiles refer to categories by numeric `cat_ids`. See the SafeDNS category reference.

Application IDs

Application IDs can be retrieved through:

```
GET /app_aware/application/
```

Allow/deny-list storage

Shield stores allowlist and denylist domains as hashes. The original domain names cannot be retrieved or enumerated through the API after submission.

Clients that need to inspect or synchronize these lists must retain their own copy of the original domains. Update and delete requests still use the original domain name; Shield calculates the corresponding hash internally.

Endpoint summary

Group	Method	Path	Purpose
Initialization	`POST`	`/init/`	Create the initial configuration in one request
Profiles	`GET`	`/profiles/`	Retrieve all profiles
Profiles	`POST`	`/profiles/`	Create a profile
Profiles	`GET`	`/profiles/{profile_id}`	Retrieve one profile
Profiles	`PATCH`	`/profiles/{profile_id}`	Update a profile and replace selected collections
Profile applications	`GET`	`/profile/{profile_id}/app_aware`	Retrieve application rules for a profile
Profile applications	`POST`	`/profile/{profile_id}/app_aware`	Add one application rule
Profile applications	`POST`	`/profile/{profile_id}/app_aware/batch`	Add several application rules with one status
Profile applications	`DELETE`	`/profile/{profile_id}/app_aware/{app_id}`	Delete an application rule

Group	Method	Path	Purpose
Application catalogue	`GET`	`/app_aware/application/`	Retrieve all available applications
Block pages	`GET`	`/blockpage/`	Retrieve all block-page records
Block pages	`POST`	`/blockpage/`	Create a block-page record
Block pages	`GET`	`/blockpage/{page_id}`	Retrieve one block-page record
Block pages	`PATCH`	`/blockpage/{page_id}`	Update a block-page record
Block pages	`DELETE`	`/blockpage/{page_id}`	Delete a block-page record
Domain lists	`POST`	`/profile/{profile_id}/bw_list`	Add one allowlist or denylist domain
Domain lists	`POST`	`/profile/{profile_id}/bw_list/batch`	Add several allowlist or denylist domains
Domain lists	`PATCH`	`/profile/{profile_id}/bw_list/{domain}`	Change a stored domain rule
Domain lists	`DELETE`	`/profile/{profile_id}/bw_list/{domain}`	Delete a stored domain rule
IPv4 networks	`GET`	`/net/`	Retrieve all IPv4 assignments
IPv4 networks	`POST`	`/net/`	Create an IPv4 assignment
IPv4 networks	`GET`	`/net/{int_ip}`	Retrieve an IPv4 assignment
IPv4 networks	`PATCH`	`/net/{int_ip}`	Update an IPv4 assignment
IPv4 networks	`DELETE`	`/net/{int_ip}`	Delete an IPv4 assignment
IPv6 networks	`GET`	`/net6/`	Retrieve all IPv6 assignments
IPv6 networks	`POST`	`/net6/`	Create an IPv6 assignment
IPv6 networks	`GET`	`/net6/{ipv6}`	Retrieve an IPv6 assignment
IPv6 networks	`PATCH`	`/net6/{ipv6}`	Update an IPv6 assignment
IPv6 networks	`DELETE`	`/net6/{ipv6}`	Delete an IPv6 assignment
NAPT	`POST`	`/napt/`	Create a NAPT assignment
NAPT	`POST`	`/napt/batch`	Create several NAPT assignments
NAPT	`GET`	`/napt/{int_ip}`	Retrieve all NAPT assignments for an IPv4 address

Group	Method	Path	Purpose
NAPT	`GET`	`/napt/{int_ip}/{lower_port_bound}/{upper_port_bound}`	Retrieve one NAPT assignment
NAPT	`PATCH`	`/napt/{int_ip}/{lower_port_bound}/{upper_port_bound}`	Update one NAPT assignment
NAPT	`DELETE`	`/napt/{int_ip}/{lower_port_bound}/{upper_port_bound}`	Delete one NAPT assignment
NAPT	`DELETE`	`/napt/batch`	Delete several NAPT assignments
Schedules	`GET`	`/profile/{profile_id}/schedule`	Retrieve a profile schedule
Schedules	`POST`	`/profile/{profile_id}/schedule`	Create a profile schedule
Schedules	`PATCH`	`/profile/{profile_id}/schedule`	Update a profile schedule
Schedules	`DELETE`	`/profile/{profile_id}/schedule`	Delete a profile schedule

Data models

Profile object

```
{
  "id": 1,
  "page_id": 1,
  "white_list_only": false,
  "empty_dns_answer": false,
  "appaware_priority_over_bwlists": false
}
```

Field	Type	Description
`id`	integer	Profile identifier. Required on creation.
`page_id`	integer or `null`	Assigned block-page identifier.
`white_list_only`	boolean	Allows only allowlisted domains and explicitly allowed applications. Default: `false`.
`empty_dns_answer`	boolean	Returns an empty DNS answer instead of a block-page response for blocked domains. Default: `false`.

Field	Type	Description
`appaware_priority_over_bwlists`	boolean	Gives application rules priority over domain allow/deny lists. Default: `false`.

Profile with categories and applications

```
{
  "profile": {
    "id": 1,
    "page_id": 1,
    "white_list_only": false,
    "empty_dns_answer": false,
    "appaware_priority_over_bwlists": false
  },
  "cat_ids": [3, 4, 12],
  "apps": [
    {
      "app_id": 12,
      "status": "allow"
    }
  ]
}
```

`cat_ids` contains filtering category IDs. `apps` contains application rules.

Application rule

```
{
  "app_id": 12,
  "status": "allow"
}
```

Field	Type	Description
`app_id`	integer	Application identifier.
`status`	string	`allow` or `deny`. Default in the schema: `deny`.

Behavior:

- `deny` blocks domains associated with the application.

- `allow` permits resolution of application-associated domains even when category rules would otherwise block them.

Application-rule response

```
{
  "profile_id": 1,
  "app_id": 12,
  "status": "allow"
}
```

Block-page object

```
{
  "id": 1,
  "type": 0
}
```

Type	Meaning
`0`	Default block page
`1`	Custom block page

Custom block-page content is configured outside the REST API.

IPv4 network object

```
{
  "ip": "100.100.100.101",
  "profile_id": 1,
  "prefix_len": 32
}
```

Field	Type	Constraints
`ip`	IPv4 string	Dotted-decimal IPv4 address or network address
`profile_id`	integer	Assigned filtering profile
`prefix_len`	integer	`10` through `32`; default `32`

IPv6 network object

```
{
  "ip": "2001:db8:100::",
  "profile_id": 1,
  "prefix_len": 64
}
```

Field	Type	Constraints
`ip`	IPv6 string	IPv6 address or network address
`profile_id`	integer	Assigned filtering profile
`prefix_len`	integer	`16` through `128`; default `128`

NAPT object

```
{
  "ip": "203.0.113.10",
  "lower_port_bound": 10000,
  "upper_port_bound": 19999,
  "profile_id": 1
}
```

Field	Type	Constraints
`ip`	IPv4 string	Dotted-decimal IPv4 address
`lower_port_bound`	integer	`0` through `65535`; default `0`
`upper_port_bound`	integer	`0` through `65535`; default `65535`
`profile_id`	integer	Assigned filtering profile

Port boundaries are inclusive.

Schedule object

```
{
  "target_profile_id": 2,
  "periods": [
    {
      "start": {
        "day": "monday",
        "time": "18:00"
      },
      "end": {
```

```
        "day": "tuesday",
        "time": "08:00"
      }
    }
  ]
}
```

`day` accepts either a lowercase weekday name or an integer:

Integer	Weekday
`0`	Monday
`1`	Tuesday
`2`	Wednesday
`3`	Thursday
`4`	Friday
`5`	Saturday
`6`	Sunday

Times use `HH:MM` and must be divisible into 30-minute intervals. `00:00` and `24:00` are supported day boundaries.

Initialization

Initialize configuration

```
POST /init/
```

Creates the initial Shield configuration from a single request containing block pages, profiles, domain rules, NAPT assignments, IPv4 networks, and IPv6 networks.

Request body

All six top-level arrays are required, including arrays that are empty.

```
{
  "blockpages": [
    {
      "id": 1,
      "type": 0
```

```
    }
  ],
  "profiles": [
    {
      "profile": {
        "id": 1,
        "page_id": 1,
        "white_list_only": false,
        "empty_dns_answer": false,
        "appaware_priority_over_bwlists": false
      },
      "cat_ids": [3, 4, 12],
      "apps": [
        {
          "app_id": 12,
          "status": "deny"
        }
      ]
    }
  ],
  "bw_lists": [
    {
      "profile_id": 1,
      "type": "deny",
      "domains": [
        "example1.com",
        "example2.com"
      ]
    }
  ],
  "napts": [],
  "nets": [
    {
      "ip": "100.100.100.101",
      "profile_id": 1,
      "prefix_len": 32
    }
  ],
  "nets6": []
}
```

Success response

```
204 No Content
```

Profiles

Retrieve all profiles

```
GET /profiles/
```

Returns all profiles, including profile settings, category IDs, and application rules.

Success response

```
200 OK
```


```
[
  {
    "profile": {
      "id": 1,
      "page_id": 1,
      "white_list_only": false,
      "empty_dns_answer": false,
      "appaware_priority_over_bwlists": false
    },
    "cat_ids": [3, 4, 12],
    "apps": [
      {
        "app_id": 12,
        "status": "deny"
      }
    ]
  }
]
```


Create a profile

```
POST /profiles/
```

Request body

```
{
  "profile": {
    "id": 2,
    "page_id": 1,
    "white_list_only": false,
    "empty_dns_answer": false,
    "appaware_priority_over_bwlists": false
  },
  "cat_ids": [3, 4, 12, 13],
  "apps": [
    {
      "app_id": 12,
      "status": "allow"
    },
    {
      "app_id": 93,
      "status": "deny"
    }
  ]
}
```

Success response

```
201 Created
```

The response body contains the created profile with its categories and application rules.

Retrieve a profile

```
GET /profiles/{profile_id}
```

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier

Success response

```
200 OK
```

The response body uses the **Profile with categories and applications** model.

Update a profile

```
PATCH /profiles/{profile_id}
```

Updates profile-level settings and optionally replaces the profile's complete category or application collections.

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier

Request body

All top-level fields are optional. Include only the areas to be changed.

```
{
  "profile": {
    "page_id": 2,
    "white_list_only": false,
    "empty_dns_answer": true,
    "appaware_priority_over_bwlists": true
  },
  "cat_ids": [3, 4, 12, 66],
  "apps": [
    {
      "app_id": 12,
      "status": "allow"
    }
  ]
}
```

Collection replacement rules

- When `cat_ids` is present, it replaces the complete existing category collection.
- `"cat_ids": []` removes all category assignments.
- When `apps` is present, it replaces the complete existing application-rule collection.
- `"apps": []` removes all application rules.
- Omitted collections remain unchanged.

Success response

```
200 OK
```

The response body contains the updated profile with its current categories and application rules.

Application catalogue and profile application rules

Retrieve all available applications

```
GET /app_aware/application/
```

Returns the application catalogue used to obtain valid `app_id` values.

Success response

```
200 OK
```

```
[
  {
    "id": 12,
    "name": "Application name"
  }
]
```

Retrieve all application rules for a profile

```
GET /profile/{profile_id}/app_aware
```

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier

Success response

```
200 OK
```

```
[
  {
    "profile_id": 1,
```

```
    "app_id": 12,
    "status": "allow"
  },
  {
    "profile_id": 1,
    "app_id": 93,
    "status": "deny"
  }
]
```

Add one application rule

```
POST /profile/{profile_id}/app_aware
```

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier

Request body

```
{
  "app_id": 12,
  "status": "allow"
}
```

`status` may be `allow` or `deny`. The schema default is `deny`, but clients should set the value explicitly.

Success response

```
201 Created
```

```
{
  "profile_id": 1,
  "app_id": 12,
  "status": "allow"
}
```

Add several application rules

```
POST /profile/{profile_id}/app_aware/batch
```

All submitted applications receive the same status.

Request body

```
{
  "app_ids": [1, 12, 93],
  "status": "deny"
}
```

Success response

```
201 Created
```

```
[
  {
    "profile_id": 1,
    "app_id": 1,
    "status": "deny"
  },
  {
    "profile_id": 1,
    "app_id": 12,
    "status": "deny"
  },
  {
    "profile_id": 1,
    "app_id": 93,
    "status": "deny"
  }
]
```

Delete an application rule

```
DELETE /profile/{profile_id}/app_aware/{app_id}
```

The current OpenAPI document names the final parameter `app_aware_id`, but the API uses the application ID itself.

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier

Parameter	Type	Description
`app_id`	integer	Application identifier

Success response

```
204 No Content
```

There is no request body.

To replace or change the complete application configuration predictably, use `PATCH /profiles/{profile_id}` with the full desired `apps` collection.

Block pages

Retrieve all block pages

```
GET /blockpage/
```

Success response

```
200 OK
```

```
[
  {
    "id": 1,
    "type": 0
  },
  {
    "id": 2,
    "type": 1
  }
]
```

Create a block page

```
POST /blockpage/
```

Request body

```
{
  "id": 2,
  "type": 1
}
```

`type` must be `0` or `1`.

Success response

```
201 Created
```

```
{
  "id": 2,
  "type": 1
}
```

Retrieve a block page

```
GET /blockpage/{page_id}
```

Path parameters

Parameter	Type	Description
`page_id`	integer	Block-page identifier

Success response

```
200 OK
```

The response body contains a block-page object.

Update a block page

```
PATCH /blockpage/{page_id}
```

Request body

```
{
  "type": 0
}
```

Success response

```
200 OK
```

The response body contains the updated block-page object.

Delete a block page

```
DELETE /blockpage/{page_id}
```

Success response

```
204 No Content
```

There is no request body.

Domain allowlists and denylists

Shield stores submitted domains as hashes. There is no GET operation for listing the original domains in a profile's allowlist or denylist.

Supported rule types are:

- `allow`
- `deny`

Clients should always send `type` explicitly.

Add one domain

```
POST /profile/{profile_id}/bw_list
```

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier

Request body

```
{
  "type": "allow",
  "domain": "example.com"
}
```

Success response

```
201 Created
```

The API returns the stored database representation:

```
{
  "profile_id": 1,
  "domain_hash": 123456789,
  "cat_id": 0
}
```

Add several domains

```
POST /profile/{profile_id}/bw_list/batch
```

Request body

```
{
  "type": "deny",
  "domains": [
    "example1.com",
    "example2.com",
    "example3.com"
  ]
}
```

Success response

```
201 Created
```

The response is an array of stored database representations.

Update a domain rule

```
PATCH /profile/{profile_id}/bw_list/{domain}
```

Changes the allow/deny state of a stored domain rule.

Path parameters

Parameter	Type	Description

`profile_id`	integer	Profile identifier
`domain`	string	Original submitted domain

Request body

The current API schema includes the domain and profile ID in the body as well as the path:

```
{
  "type": "allow",
  "domain": "example.com",
  "profile_id": 1
}
```

Success response

```
200 OK
```

The response uses the stored domain-rule representation containing `profile_id`, `domain_hash`, and `cat_id`.

Delete a domain rule

```
DELETE /profile/{profile_id}/bw_list/{domain}
```

Path parameters

Parameter	Type	Description
`profile_id`	integer	Profile identifier
`domain`	string	Original submitted domain

Success response

```
204 No Content
```

There is no request body.

IPv4 networks

Supported prefix lengths are `/10` through `/32`.

Retrieve all IPv4 assignments

```
GET /net/
```

Success response

```
200 OK
```

```
[
  {
    "ip": "100.100.100.101",
    "profile_id": 1,
    "prefix_len": 32
  },
  {
    "ip": "100.100.101.0",
    "profile_id": 2,
    "prefix_len": 24
  }
]
```

Create an IPv4 assignment

```
POST /net/
```

Request body

Host assignment:

```
{
  "ip": "100.100.100.101",
  "profile_id": 1,
  "prefix_len": 32
}
```

Subnet assignment:

```
{
  "ip": "100.100.101.0",
  "profile_id": 1,
  "prefix_len": 24
}
```

Success response

```
201 Created
```

The response body contains the created IPv4 network object.

Retrieve an IPv4 assignment

```
GET /net/{int_ip}
```

`int_ip` is the unsigned integer representation of the address or network address.

Example:

```
GET /net/1684301056
```

retrieves the record identified by `100.100.101.0`.

Success response

```
200 OK
```

The response body contains an IPv4 network object.

Update an IPv4 assignment

```
PATCH /net/{int_ip}
```

Request body

```
{
  "ip": "100.100.101.0",
  "profile_id": 2,
  "prefix_len": 24
}
```

Success response

```
200 OK
```

The response body contains the updated IPv4 network object.

Delete an IPv4 assignment

```
DELETE /net/{int_ip}
```

Success response

```
204 No Content
```

There is no request body.

IPv6 networks

Supported prefix lengths are `/16` through `/128`.

Retrieve all IPv6 assignments

```
GET /net6/
```

Success response

```
200 OK
```

```
[
  {
    "ip": "2001:db8:100::",
    "profile_id": 1,
    "prefix_len": 64
  }
]
```

Create an IPv6 assignment

```
POST /net6/
```

Request body

```
{
  "ip": "2001:db8:100::",
  "profile_id": 1,
  "prefix_len": 64
}
```

Success response

```
201 Created
```

The response body contains the created IPv6 network object.

Retrieve an IPv6 assignment

```
GET /net6/{ipv6}
```

Path parameters

Parameter	Type	Description
`ipv6`	IPv6 string	IPv6 address or network address identifying the record

Success response

```
200 OK
```

The response body contains an IPv6 network object.

Update an IPv6 assignment

```
PATCH /net6/{ipv6}
```

Request body

```
{
  "ip": "2001:db8:100::",
  "profile_id": 2,
  "prefix_len": 64
}
```

Success response

```
200 OK
```

The response body contains the updated IPv6 network object.

Delete an IPv6 assignment

```
DELETE /net6/{ipv6}
```

Success response

```
204 No Content
```

There is no request body.

NAPT assignments

NAPT assignments map inclusive source-port ranges on a shared IPv4 address to filtering profiles.

Port constraints:

- Minimum: `0`
- Maximum: `65535`
- Default lower bound: `0`
- Default upper bound: `65535`
- Both boundaries are included in the assignment.

Individual resource paths use integer `int_ip`. Request and response bodies use dotted-decimal `ip` strings.

Create a NAPT assignment

```
POST /napt/
```

Request body

```
{
  "ip": "203.0.113.10",
  "lower_port_bound": 10000,
  "upper_port_bound": 19999,
  "profile_id": 1
}
```

Success response

```
201 Created
```

The response body contains the created NAPT object.

Create several NAPT assignments

```
POST /napt/batch
```

Request body

```
[
  {
    "ip": "203.0.113.10",
    "lower_port_bound": 10000,
    "upper_port_bound": 19999,
    "profile_id": 1
  },
  {
    "ip": "203.0.113.10",
    "lower_port_bound": 20000,
    "upper_port_bound": 29999,
    "profile_id": 2
  }
]
```

Success response

```
201 Created
```

The response is an array of created NAPT objects.

Retrieve all NAPT assignments for an IPv4 address

```
GET /napt/{int_ip}
```

Example:

```
GET /napt/3405803786
```

`3405803786` is the integer representation of `203.0.113.10`.

Success response

```
200 OK
```

```
[
  {
    "ip": "203.0.113.10",
```

```
    "lower_port_bound": 10000,
    "upper_port_bound": 19999,
    "profile_id": 1
  },
  {
    "ip": "203.0.113.10",
    "lower_port_bound": 20000,
    "upper_port_bound": 29999,
    "profile_id": 2
  }
]
```

Retrieve one NAPT assignment

```
GET /napt/{int_ip}/{lower_port_bound}/{upper_port_bound}
```

Example:

```
GET /napt/3405803786/10000/19999
```

Success response

```
200 OK
```

The response body contains one NAPT object.

Update a NAPT assignment

```
PATCH /napt/{int_ip}/{lower_port_bound}/{upper_port_bound}
```

The path identifies the current record. The body contains the desired resulting values.

Request body

```
{
  "ip": "203.0.113.10",
  "lower_port_bound": 10000,
  "upper_port_bound": 19999,
  "profile_id": 2
}
```

Success response

```
200 OK
```

The response body contains the updated NAPT object.

Delete one NAPT assignment

```
DELETE /napt/{int_ip}/{lower_port_bound}/{upper_port_bound}
```

Success response

```
204 No Content
```

There is no request body.

Delete several NAPT assignments

```
DELETE /napt/batch
```

The batch-deletion request uses dotted-decimal `ip` strings, not integer `int_ip` values. It does not include `profile_id`.

Request body

```
[
  {
    "ip": "203.0.113.10",
    "lower_port_bound": 10000,
    "upper_port_bound": 19999
  },
  {
    "ip": "203.0.113.10",
    "lower_port_bound": 20000,
    "upper_port_bound": 29999
  }
]
```

Success response

```
204 No Content
```

Profile schedules

A schedule switches the profile identified by `{profile_id}` to `target_profile_id` during configured periods.

The source profile and target profile must be different.

Day and time values

`day` accepts either:

- A lowercase weekday name: `monday` through `sunday`.
- An integer from `0` through `6`, where `0` is Monday and `6` is Sunday.

`time` uses `HH:MM` format. Valid times fall on 30-minute boundaries. `00:00` and `24:00` are supported.

Retrieve a schedule

```
GET /profile/{profile_id}/schedule
```

Success response

```
200 OK
```

```
{
  "target_profile_id": 2,
  "periods": [
    {
      "start": {
        "day": "monday",
        "time": "18:00"
      },
      "end": {
        "day": "tuesday",
        "time": "08:00"
      }
    }
  ]
}
```

Create a schedule

```
POST /profile/{profile_id}/schedule
```

Request body

```
{
  "target_profile_id": 2,
  "periods": [
    {
      "start": {
        "day": "monday",
        "time": "18:00"
      },
      "end": {
        "day": "tuesday",
        "time": "08:00"
      }
    },
    {
      "start": {
        "day": 5,
        "time": "00:00"
      },
      "end": {
        "day": 5,
        "time": "24:00"
      }
    }
  ]
}
```

Success response

```
201 Created
```

The response body contains the created schedule.

Update a schedule

```
PATCH /profile/{profile_id}/schedule
```

The request may update the target profile, the periods, or both.

Change the target profile

```
{
  "target_profile_id": 3
}
```

When only `target_profile_id` is supplied, it is changed for all existing periods.

Replace all periods

```
{
  "periods": [
    {
      "start": {
        "day": "friday",
        "time": "18:00"
      },
      "end": {
        "day": "monday",
        "time": "08:00"
      }
    }
  ]
}
```

When `periods` is supplied, all previously configured periods are deleted and replaced by the submitted collection.

Success response

```
200 OK
```

The response body contains the updated schedule.

Delete a schedule

```
DELETE /profile/{profile_id}/schedule
```

Success response

```
204 No Content
```

There is no request body.

Complete curl examples

Retrieve profiles

```
curl \
  --request GET \
  --header "Authorization: Bearer <token>" \
  --header "Accept: application/json" \
  "<base-url>/profiles/"
```

Create an IPv4 assignment

```
curl \
  --request POST \
  --header "Authorization: Bearer <token>" \
  --header "Content-Type: application/json" \
  --data '{
    "ip": "100.100.100.101",
    "profile_id": 1,
    "prefix_len": 32
  }' \
  "<base-url>/net/"
```

Replace profile categories and application rules

```
curl \
  --request PATCH \
  --header "Authorization: Bearer <token>" \
  --header "Content-Type: application/json" \
  --data '{
    "cat_ids": [3, 4, 12, 66],
    "apps": [
      {
        "app_id": 12,
        "status": "allow"
      },
      {
        "app_id": 93,
```

```
        "status": "deny"
      }
    ]
}' \
"<base-url>/profiles/1"
```

Revision #7
Created 2026-05-26 13:15:11 UTC by Andrew Lem
Updated 2026-07-22 15:07:37 UTC by Andrew Lem