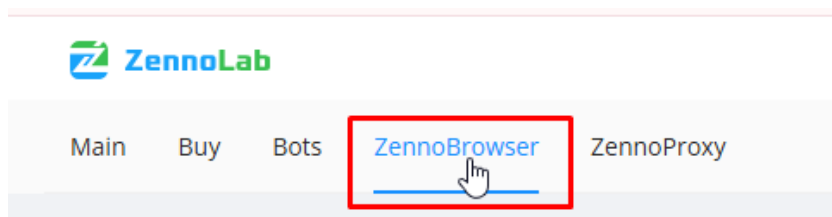


ZennoBrowser Public API

Getting Started with the ZennoBrowser API

The API Token is required to execute all requests. It can be obtained as follows:

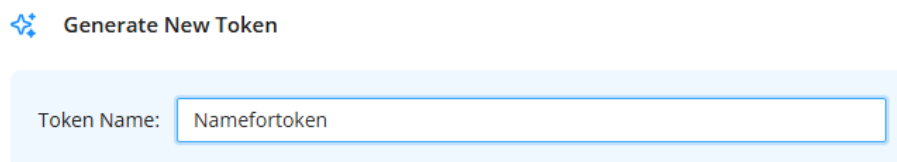
1. Log in to UserArea2 with active ZennoBrowser subscription;
2. Open "ZennoBrowser" section at the top of the main page;



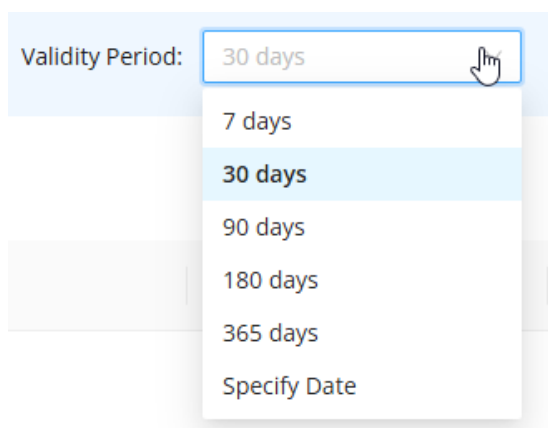
3. Click the "API Control" link;



4. Specify token name (minimum 3 characters);



5. Select the period which this token will be valid for (7 days, 30 days, 60 days, etc.);



6. Click "Generate token";

Generate New Token

Token Name: Validity Period:

7. Copy the generated token. After closing the window, it is not possible to copy the token again, so be sure to save the generated token.

Token Generated

Be sure to save the token—you will not be able to access it after closing this window.

8. Use the generated token to execute requests.

Note:

1. After deleting the token, it will continue functioning within 1.5-2 hours;
2. After token has expired, it will cease to function;
3. If you have lost a previously created token, you can create a new one.

Work with WorkSpaces

Getting accessible workspaces

Description: The operation retrieves the list of workspaces accessible to the current user. This information is required to execute most subsequent queries.

Request parameters:

Parameter	Type	Format	Default	Description
<code>start</code>	integer	int32	0	Start index for pagination
<code>total</code>	integer	int32	1000	Maximum number of results to return
<code>id</code>	integer	int64	(empty)	Optional workspace ID filter
<code>name</code>	string		(empty)	Optional workspace name filter
<code>roles</code>	string		(empty)	Optional roles filter

Example request:

GET

CURL:

```
curl
'http://localhost:8160/v1/workspaces?start=0&total=1000&id=&name=&roles='
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Get,
    RequestUri = new Uri(
"http://localhost:8160/v1/workspaces?start=0&total=1000&id=&name=&roles="),
    Headers =
    {
```

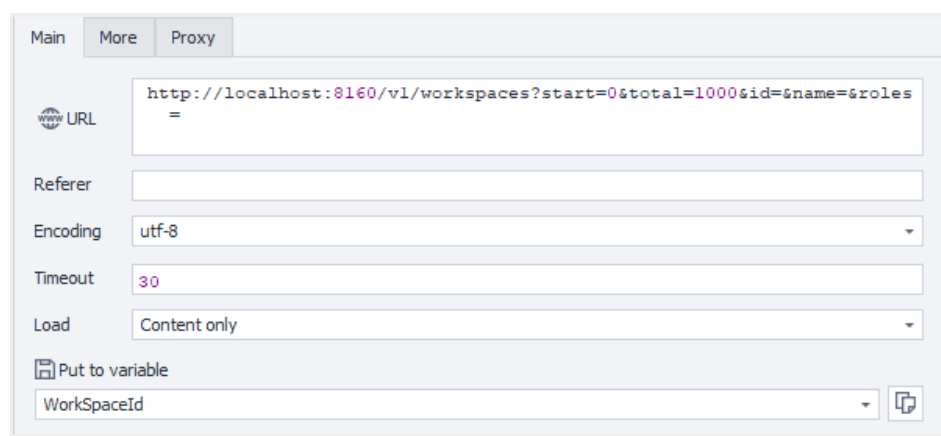
```

        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

<http://localhost:8160/v1/workspaces?start=0&total=1000&id=&name=&roles=>



Main More Proxy

URL `http://localhost:8160/v1/workspaces?start=0&total=1000&id=%name=%roles=`

Referer

Encoding utf-8

Timeout 30

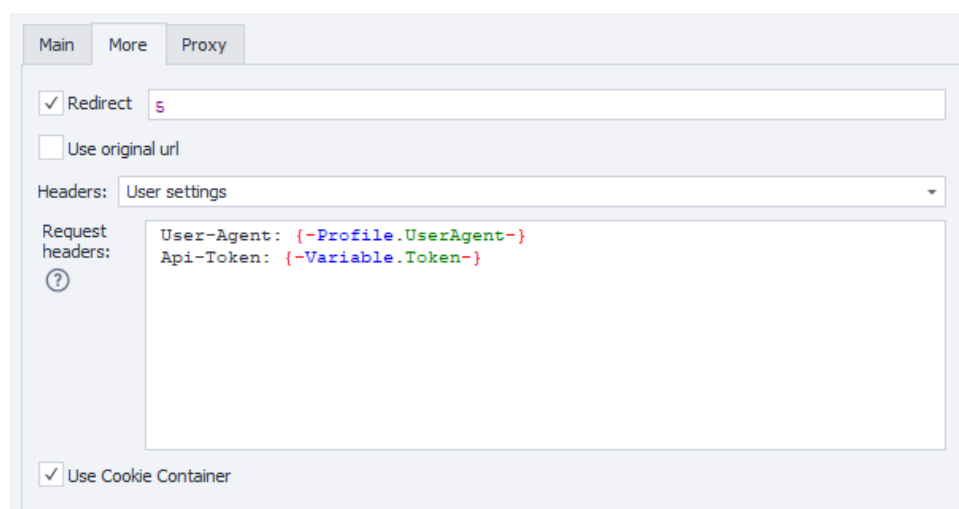
Load Content only

Put to variable WorkspaceId

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.



Main More Proxy

☒ Redirect 5

☐ Use original url

Headers: User settings

Request headers: `User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre> { "totalCount": 1, "items": [{ "workspaceId": 1, "name": null, "role": "Owner" }] } </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Folder Management

Getting folder list

Description: This retrieves a list of folders available in the specified workspace. Optionally, it is possible to limit the number of results, specify filters, or define sorting conditions.

Request parameters:

Parameter	Type	Format	Default	Description
<code>workspaceId</code>	integer	int64	-1	ID of the workspace where the profile folder will be created. Defaults to -1 if not specified.
<code>start</code>	integer	int32	0	Index of the first folder (starting from zero).
<code>total</code>	integer	int32	1000	Maximum number of folders to retrieve.
<code>id</code>	string	uuid	(empty)	Unique folder identifier (optional filter).
<code>name</code>	string		(empty)	Folder name (optional filter).
<code>sorting</code>	string		(empty)	Sorting condition: <code>Id</code> , <code>Name</code> , <code>FolderType</code> , <code>Quantity</code> (optional filter).
<code>location</code>	string (enum)		Local	Folder location: " <code>Local</code> " or " <code>Cloud</code> ".

Example request:

GET

CURL :

```
curl
'http://localhost:8160/v1/profile_folders?workspaceId=-1&start=0&total=1000&id=&name=&sorting=&location=' \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
```

```

var request = new HttpRequestMessage
{
    Method = HttpMethod.Get,
    RequestUri = new Uri (
"http://localhost:8160/v1/profile_folders?workspaceId=-1&start=0&total=
1000&id=&name=&sorting=&location="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

http://localhost:8160/v1/profile_folders?workspaceId=-1&start=0&total=1000&id=&name=&sorting=&location=

The screenshot shows a web client interface with the following fields and values:

- URL:** `http://localhost:8160/v1/profile_folders?workspaceId=-1&start=0&total=1000&id=&name=&sorting=&location=`
- Referer:** (empty)
- Encoding:** `utf-8`
- Timeout:** `30`
- Load:** `Content only`
- Put to variable:** `Response`

Additionally:

User-Agent: {-Profile.UserAgent-}
 Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect
5

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre> { "totalCount": 1, "items": [{ "id": "123e4567-e89b-12d3-a456-426614174000", "name": null, "location": "Local", "quantity": 1 }] } </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Creating new profile folder

Description: This method creates a profile folder. The caller must ensure that the specified workspace ID and location are valid.

Request parameters:

Parameter	Type	Format	Default	Description
<code>name</code>	string			Name of the profile folder. Cannot be null or empty.
<code>workspaceId</code>	integer	int64	-1	ID of the workspace where the profile folder will be created. Defaults to -1 if not specified.
<code>location</code>	string (enum)		Local	The location where the profile folder will be created. Folder location: " <code>Local</code> " or " <code>Cloud</code> ".

Example request:

POST

CURL:

```
curl
'http://localhost:8160/v1/profile_folders?name=&workspaceId=-1&location=Local' \
--request POST \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri(
"http://localhost:8160/v1/profile_folders?name=&workspaceId=-1&location=Local"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

}

Cube:

http://localhost:8160/v1/profile_folders?name=&workspaceId=-1&location=Local

The screenshot shows the 'Main' tab of a web client interface. It contains the following fields and settings:

- URL:** `http://localhost:8160/v1/profile_folders?name=TestFolder&workspaceId=-1&location=Local`
- Referer:** (Empty text field)
- Encoding:** `utf-8` (Dropdown menu)
- Timeout:** `30` (Text field)
- Data:** (Large empty text area)
- Data type:** `urlencoded` (Dropdown menu)
- Load:** `Content only` (Dropdown menu)
- Put to variable:** `Response` (Dropdown menu with a copy icon)

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

The screenshot shows the 'More' tab of the same web client interface. It contains the following fields and settings:

- Redirect:** ☒ `5` (Text field)
- Use original url:** ☐ (Checkbox)
- Headers:** `User settings` (Dropdown menu)
- Request headers:** `User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}` (Text area)
- Use Cookie Container:** ☒ (Checkbox)

Response API:

Response code	Result	Response
200 OK	Success	id new profile folder (123e4567-e89b-12d3-a456-426614174000)
500 Error	Internal Server Error	{ "message": null }

Updating profile folder

Description: This method can be used to update the profile folder name.

Request parameters:

Parameter	Type	Format	Default	Description
profileFolderId	string	uuid		Unique identifier of the profile folder to update.
name	string			The new name for the profile folder.
workspaceId	integer	int64	-1	The identifier of the workspace associated with the profile folder. Defaults to -1 if not specified.

Note: The ID of the edited profile folder should be specified inside curly braces{ } in the URL path.

Example request:

PUT

CURL :

```
curl
'http://localhost:8160/v1/profile_folders/{profileFolderId}?name=&workspaceId=-1' \
--request PUT \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Put,
    RequestUri = new Uri(
"http://localhost:8160/v1/profile_folders/{profileFolderId}?name=&wo
rkspaceId=-1"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

http://localhost:8160/v1/profile_folders/{profileFolderId}?name=&workspaceId=-1

The screenshot shows the Postman application interface with the 'Main' tab selected. A red box highlights the 'Put' method dropdown. The URL field contains the request URL: `http://localhost:8160/v1/profile_folders/123e4567-e89b-12d3-a456-426614174000?name=FolderRename&workspaceId=-1`. The 'Referer' field is empty. The 'Encoding' dropdown is set to 'utf-8'. The 'Timeout' field is set to '30'. The 'Data' section is empty. The 'Data type' dropdown is set to 'urlencoded'. The 'Load' dropdown is set to 'Content only'. The 'Put to variable' dropdown is set to 'Response'.

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

The screenshot shows a configuration window with tabs for 'Main', 'More', and 'Proxy'. The 'Main' tab is active. It contains a 'Redirect' checkbox checked with a value of '5', an unchecked 'Use original url' checkbox, a 'Headers' dropdown menu set to 'User settings', and a 'Request headers' section. The 'Request headers' section contains two entries: 'User-Agent: {-Profile.UserAgent-}' and 'Api-Token: {-Variable.Token-}'. A 'Use Cookie Container' checkbox is checked at the bottom.

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting profile folder

Description: This method is used for deleting a profile folder. The profileFolderId value must match the existing folder in the specified workspace.

Request parameters:

Parameter	Type	Format	Default	Description
profileFolderId	string	uuid		Unique identifier of the profile folder to delete.

<code>workspaceId</code>	integer	int64	-1	The identifier of the workspace associated with the profile folder. Defaults to -1 if not specified.
--------------------------	---------	-------	----	--

Note: The ID of the edited profile folder should be specified inside curly braces { } in the URL path.

Example request:

DELETE

CURL :

```
curl
'http://localhost:8160/v1/profile_folders/{profileFolderId}?workspaceId=-1' \
--request DELETE \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new Uri(
"http://localhost:8160/v1/profile_folders/%7BprofileFolderId%7D"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube :

http://localhost:8160/v1/profile_folders/{profileFolderId}?workspaceId=-1

Main More Proxy

Delete

URL `http://localhost:8160/v1/profile_folders/123e4567-e89b-12d3-a456-426614174000?name=FolderRename&workspaceId=-1`

Referer

Encoding `utf-8`

Timeout `30`

Data

Data type `urlencoded`

Load `Content only`

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main More Proxy

☒ Redirect `5`

☐ Use original url

Headers: `User settings`

Request headers:

`User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

Response API:

Response code	Result	Response
<code>200 OK</code>	<code>Success</code>	

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------------	--

Profile Management

Getting accessible browser profiles

Description: This method retrieves a list of profiles with their attributes from the specified workspace and the [folder](#). It is possible to optionally filter results by ID, name, tags, sorting.

Request parameters:

Parameter	Type	Format	Default	Description
<code>workspaceId</code>	integer	int64	-1	Workspace identifier. -1 means the default workspace.
<code>start</code>	integer	int32	0	Zero-based index of the first profile folder to retrieve. Defaults to 0.
<code>total</code>	integer	int32	1000	Maximum number of results to return
<code>folderId</code>	string	uuid	(empty)	Folder identifier for filtering (optional).
<code>id</code>	string	uuid	(empty)	Profile identifier for filtering (optional).
<code>name</code>	string		(empty)	Profile name for filtering (optional).
<code>tags</code>	string		(empty)	Tags for filtering (optional). Multiple tags can be separated by spaces.
<code>sorting</code>	string		(empty)	Sorting parameters (optional). <code>Id</code> , <code>ContainerId</code> , <code>Name</code> , <code>CreationTime</code> , <code>User.Sex</code> , <code>User.Name</code> , <code>User.Surname</code> , <code>User.Age</code> , <code>User.Country</code> , <code>OperationSystem.Name</code> , <code>OperationSystem.Version</code> , <code>Brand.Name</code> , <code>Brand.Version</code> , <code>Proxy</code> , <code>Proxy.Name</code> , <code>ProfileState.DisplayStatus</code> , <code>ProfileState.Description</code> , <code>Status</code> , <code>OperationalState.OperationStatus</code> , <code>OperationalState.CanChangeOperationStatus</code> , <code>UsedTime</code> , <code>LastStartTime</code> , <code>LastErrorMessage</code> , <code>Tags</code> , <code>Tags.Id</code> , <code>Notes</code>

Example request:

GET

CURL:

```
curl
```

```
'http://localhost:8160/v1/profiles?workspaceId=-1&start=0&total=1000
&folderId=&id=&name=&tags=&sorting=' \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Get,
    RequestUri = new Uri(
"http://localhost:8160/v1/profiles?workspaceId=-1&start=0&total=1000
&folderId=&id=&name=&tags=&sorting="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

```
http://localhost:8160/v1/profiles?workspaceId=-1&start=0&total=1000&
folderId=&id=&name=&tags=&sorting=
```

Main
More
Proxy

URL

http://localhost:8160/v1/profiles?workspaceId=-1&start=0&total=1000&folderId=&id=&name=&tags=&sorting=

Referer

Encoding
utf-8

Timeout
30

Load
Content only

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**

Main
More
Proxy

☒ Redirect
5

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre> { "totalCount": 1, "items": [{ "id": "123e4567-e89b-12d3-a456-426614174000", "name": null, "folderId": "123e4567-e89b-12d3-a456-426614174000", "creationTime": </pre>

		<pre> "2025-07-25T09:34:29.768Z", "user": { "sex": "Male", "name": null, "surname": null, "age": 1, "country": null }, "browser": { "operationSystem": { "name": null, "version": null }, "brand": { "name": null, "version": null } }, "proxy": { "id": "123e4567-e89b-12d3-a456-426614174000", "name": null }, "status": null, "usedTimeSeconds": null, "lastStartTime": null, "lastErrorMessage": null, "tags": [{ "name": null, "color": null }], "notes": null }] } </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Creating browser profile

Description: This method creates a new browser profile in the specified workspace and the folder.

It is possible to customize various parameters such as screen size, hardware emulation, language, timezone, proxy, etc..

All parameters are optional if another is not specified.

Request parameters:

Parameter	Type	Format	Default	Description
name	string			The profile name.
workspace Id	integer	int64	-1	Workspace identifier. -1 means the default workspace.
folderId	string	uuid	(empty)	The folder identifier (optional).
screen	string		(empty)	Screen configuration settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value or "width x height" to set custom screen resolution (auto, ignore or or manual screen resolution entry, such as SVGA, WSVGA, HD+, etc. The complete list of manually configurable resolutions is provided below) "SVGA" = 800×600; "0.69M3" = 960×720; "0.59M9" = 1024×576; "WSVGA" = 1024×600; "0.66MA" = 1024×640; "XGA" = 1024×768; "1152×648" = 1152×648; "XGA+" = 1152×864; "HD" = 1280×720;

				"WXGA" = 1280×768; "WXGA+" = 1280×800; "SXGA-" = 1280×960; "SXGA" = 1280×1024; "1360×768" = 1360×768; "WXGA HD" = 1366×768; "SXGA+" = 1400×1050; "WSXGA" = 1440×900; "1.56M3" = 1440×1080; "1536×864" = 1536×864; "HD+" = 1600×900; "UXGA" = 1600×1200; "WSXGA+" = 1680×1050; "1832×1392" = 1832×1392; "FHD" = 1920×1080; "WUXGA" = 1920×1200; "2.76M3" = 1920×1440; "QWXGA" = 2048×1152; "QXGA" = 2048×1536; "3.32MA" = 2304×1440; "WQHD" = 2560×1440; "WQXGA" = 2560×1600; "QSXGA" = 2560×2048; "5.18MA" = 2880×1800; "4K UHD-1" = 3840×2160; "9.44M9" = 4096×2304; "5K" = 5120×2880; "8K UHD-2" = 7680×4320;
cpu	string		(empty)	CPU configuration settings. The allowed values are: "auto" to use value from

				fingerprint, “ignore” to use system value or integer number to set custom processor core count (auto, ignore, or number of cores ("2/4", "4/8", "6/12", "8/16", "10/20", "12/24", "14/28", "16/32", "32/64", "64/128"))
memory	string		(empty)	Memory configuration settings. The allowed values are: “auto” to use value from fingerprint, “ignore” to use system value or integer number to set custom memory size (auto, ignore or memory size ("2", "4", "8", "16", "32", "64", "128"))
language	string		(empty)	Browser language settings. The allowed values are: “auto” to use value from fingerprint, “ignore” to use system value or language locale value (auto, ignore, or locale: Russian - ru English (United States) - en-US German - de French - fr Spanish - es English (United Kingdom) - en-GB
geoLocation	string		(empty)	Geographic location settings. The allowed values are: “auto” to use value from fingerprint, “ignore” to use system value. It is also possible to specify from one to seven floating point numbers that represent the following geo coordinates: Latitude, Longitude, Altitude, Accuracy, AltitudeAccuracy, Heading, Speed, Radius. Some coordinates can be skipped. For example, the following value sequence is valid: 10, , 40, , 60, 70 - means Latitude = 10, Longitude = random, Altitude = 40, Accuracy = random, AltitudeAccuracy = 60, Heading = 70, Speed = random, Radius

				= random (auto, ignore or coordinates (latitude, longitude, etc.))
timeZone	string		(empty)	Timezone settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value or custom timezone value (auto, ignore, or specific value (e.g. Europe/Moscow))
webGl	string		(empty)	WebGL configuration settings. The allowed values are: "auto: to use value from fingerprint, "ignore" to use system value. (auto or ignore)
webGpu	string		(empty)	WebGPU configuration settings. The allowed values are: "on" to enable or "off" to disable this feature.(on or off)
webRtc	string		(empty)	WebRTC configuration settings. The allowed values are: "Emulate" to use value from fingerprint, "Real" to use system value or "Hide" to disable this feature.(Emulate, Real or Hide)
domRect	string		(empty)	DOM Rectangle configuration settings. The allowed values are: "Ignore" to disable this feature or "Noise" to use dom rect noising.(Ignore or Noise)
audio	string		(empty)	Audio configuration settings. The allowed values are: "on" to enable or "off" to disable this feature. (on or off)

fonts	string		(empty)	Font configuration settings. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
battery	string		(empty)	Battery configuration settings. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
plugins	string		(empty)	Browser plugins configuration. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
proxyServerId	string	uuid	(empty)	Proxy server identifier (optional).
speechVoices	string		(empty)	Speech voices configuration. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
cookies	string		(empty)	Cookies configuration
notes	string		(empty)	Additional notes for the profile.
tags	string		(empty)	Profile tags separated by spaces. (e.g. 1 2 3 4)

Example request:

POST

CURL :

curl

```
'http://localhost:8160/v1/profiles/create?name=&workspaceId=-1&folderId=&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&domRect=&audio=&fonts=&battery=&plugins=&proxyServerId=&speechVoices=&cookies=&notes=&tags=' \
--request POST \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri(
"http://localhost:8160/v1/profiles/create?name=&workspaceId=-1&folderId=&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&domRect=&audio=&fonts=&battery=&plugins=&proxyServerId=&speechVoices=&cookies=&notes=&tags="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

<http://localhost:8160/v1/profiles/create?name=&workspaceId=-1&folderId=&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&domRect=&audio=&fonts=&battery=&plugins=&proxyServerId=&speechVoices=&cookies=¬es=&tags=>

Main More Proxy

URL `http://localhost:8160/v1/profiles/create?name=ApiProfile&workspaceId=-1&folderId=e655c070-bb03-42ef-b610-c4a223f7ce63&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&do`

Referer

Encoding `utf-8`

Timeout `30`

Data

Data type `urlencoded`

Load `Content only`

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**

Main More Proxy

☒ Redirect `5`

☐ Use original url

Headers: `User settings`

Request headers:

`User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	id new profile (123e4567-e89b-12d3-a456-426614174000)

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------	----------------------------------

Creating browser profiles (bulk)

Description: This method is used to create new multiple browser profiles in the specified workspace and the folder.

Each profile can have its own configuration, including screen size, hardware emulation, language, timezone, proxy, and other parameters.

All fields are optional if another is not specified.

Request parameters:

Parameter	Type	Format	Default	Description
count	integer	int32		Number of profiles to create
workspace Id	integer	int64	-1	Workspace identifier. -1 means the default workspace.
folderId	string	uuid	(empty)	Folder identifier (optional).
screen	string		(empty)	Screen configuration settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value or "width x height" to use custom screen resolution (auto, ignore or manual screen resolution entry, such as SVGA, WSVGA, HD+, etc. The complete list of manually configurable resolutions is provided below) "SVGA" = 800×600; "0.69M3" = 960×720; "0.59M9" = 1024×576;

				"WSVGA" = 1024×600; "0.66MA" = 1024×640; "XGA" = 1024×768; "1152×648" = 1152×648; "XGA+" = 1152×864; "HD" = 1280×720; "WXGA" = 1280×768; "WXGA+" = 1280×800; "SXGA-" = 1280×960; "SXGA" = 1280×1024; "1360×768" = 1360×768; "WXGA HD" = 1366×768; "SXGA+" = 1400×1050; "WSXGA" = 1440×900; "1.56M3" = 1440×1080; "1536×864" = 1536×864; "HD+" = 1600×900; "UXGA" = 1600×1200; "WSXGA+" = 1680×1050; "1832×1392" = 1832×1392; "FHD" = 1920×1080; "WUXGA" = 1920×1200; "2.76M3" = 1920×1440; "QWXGA" = 2048×1152; "QXGA" = 2048×1536; "3.32MA" = 2304×1440; "WQHD" = 2560×1440; "WQXGA" = 2560×1600; "QSXGA" = 2560×2048; "5.18MA" = 2880×1800;
--	--	--	--	---

				"4K UHD-1" = 3840×2160; "9.44M9" = 4096×2304; "5K" = 5120×2880; "8K UHD-2" = 7680×4320;)
cpu	string		(empty)	CPU configuration settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value or integer number to use custom processor core count (auto , ignore , or number of cores ("2/4", "4/8", "6/12", "8/16", "10/20", "12/24", "14/28", "16/32", "32/64", "64/128"))
memory	string		(empty)	Memory configuration settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value or integer number to custom memory size (auto , ignore or memory size ("2", "4", "8", "16", "32", "64", "128"))
language	string		(empty)	Browser language settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value or custom language value (auto , ignore , or locale Russian - ru English (United States) - en-US German - de French - fr Spanish - es English (United Kingdom) - en-GB
geoLocation	string		(empty)	Geographic location settings. The allowed values are: "auto" to use value from fingerprint, "ignore" to use system value. You can also specify from one to seven floating point numbers that represent the following

				geo coordinates: Latitude, Longitude, Altitude, Accuracy, AltitudeAccuracy, Heading, Speed, Radius. Some coordinates can be skipped. For example, the following sequence is valid: 10, , 40, , 60, 70 - means Latitude = 10, Longitude = random, Altitude = 40, Accuracy = random, AltitudeAccuracy = 60, Heading = 70, Speed = random, Radius = random (auto , ignore or coordinates (latitude, longitude, etc.))
timeZone	string		(empty)	Time zone settings. The allowed values are: “auto” to use value from fingerprint, “ignore” to use system value or custom timezone value (auto , ignore , or specific value (e.g. Europe/Moscow))
webGl	string		(empty)	WebGL configuration settings. The allowed values are: “auto” to use value from fingerprint, “ignore” to use system value. (auto or ignore)
webGpu	string		(empty)	WebGPU configuration settings. The allowed values are: on to enable or off to disable this feature.(on or off)
webRtc	string		(empty)	WebRTC configuration settings. The allowed values are: ‘Emulate’ to use value from fingerprint, ‘Real’ to use system value or “Hide” to disable this feature.(Emulate , Real or Hide)
domRect	string		(empty)	DOM Rectangle configuration settings. The allowed values are: “Ignore” to disable this feature or “Noise” to use dom rect noising.(Ignore or Noise)

audio	string		(empty)	Audio configuration settings. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
fonts	string		(empty)	Font configuration settings. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
battery	string		(empty)	Battery configuration settings. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
plugins	string		(empty)	Browser plugins configuration. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
proxyServerId	string	uuid	(empty)	Proxy server identifier (optional).
speechVoices	string		(empty)	Speech voices configuration. The allowed values are: “on” to enable or “off” to disable this feature. (on or off)
tags	string		(empty)	Profile tags separated by spaces or commas. (e.g. 1 2 3 4)

Example request:

POST

CURL :

```
curl
```

```
'http://localhost:8160/v1/profiles/create_bulk?count=&workspaceId=-1
&folderId=&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&web
Gl=&webGpu=&webRtc=&domRect=&audio=&fonts=&battery=&plugins=&proxySe
rverId=&speechVoices=&tags=' \
```

```
--request POST \
```

```
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri(
"http://localhost:8160/v1/profiles/create_bulk?count=&workspaceId=-1&folderId=&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&domRect=&audio=&fonts=&battery=&plugins=&proxyServerId=&speechVoices=&tags="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/profiles/create_bulk?count=&workspaceId=-1&folderId=&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&domRect=&audio=&fonts=&battery=&plugins=&proxyServerId=&speechVoices=&tags=
```

Main More Proxy

URL `http://localhost:8160/v1/profiles/create_bulk?count=5&workspaceId=-1&folderId=e655c070-bb03-42ef-b610-c4a223f7ce63&screen=&cpu=&memory=&language=&geoLocation=&timeZone=&webGl=&webGpu=&webRtc=&domRe`

Referer

Encoding `utf-8`

Timeout `30`

Data

Data type `urlencoded`

Load `Content only`

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**

Main More Proxy

☒ Redirect `s`

☐ Use original url

Headers: `User settings`

Request headers:

`User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre>id new profiles ["123e4567-e89b-12d3-a456-426614174000"]</pre>

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------	----------------------------------

Updating browser profile

Description: This method updates an existing browser profile.

Request parameters:

Parameter	Type	Format	Default	Description
profileId	string	uuid	(required)	Unique identifier of the edited profile (required).
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
folderId	string	uuid	(empty)	New folder which the profile will be moved to (optional).
name	string		(empty)	New profile name.
notes	string		(empty)	New profile notes (optional).
tags	string		(empty)	New profile tags separated by spaces (optional).

Note: The ID of the edited profile should be specified inside curly braces { } in the URL path.

Example request:

PUT

CURL :

```
curl
'http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1&folderId=a2f76986-8064-4c2f-b109-64a84b4ce137&name=NewName&notes=Note123&tags=tag1 tag2' \
--request PUT \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Put,
    RequestUri = new Uri(
"http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1&folderId=a2f76986-8064-4c2f-b109-64a84b4ce137&name=NewName&notes=Note123&tags=tag1 tag2"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1&folderId=a2f76986-8064-4c2f-b109-64a84b4ce137&name=NewName&notes=Note123&tags=tag1 tag2
```

Main More Proxy

Put

URL `http://localhost:8160/v1/profiles/c98b47b8-568e-41e1-bc2c-d9e015488d0b?workspaceId=-1&folderId=e655c070-bb03-42ef-b610-c4a223f7ce63&name=NewName¬es=Notel23&tags=tag3 tag4`

Referer

Encoding utf-8

Timeout 30

Data

Data type urlencoded

Load Content only

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main More Proxy

☒ Redirect s

☐ Use original url

Headers: User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------	----------------------------------

Deleting browser profile

Description: This method deletes an existing browser profile.

Request parameters:

Parameter	Type	Format	Default	Description
profileId	string	uuid	(required)	Unique identifier of the profile to delete (required).
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note: The profile ID should be specified inside curly braces { } in the URL path.

Example request:

DELETE

CURL :

```
curl
'http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1' \
--request DELETE \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

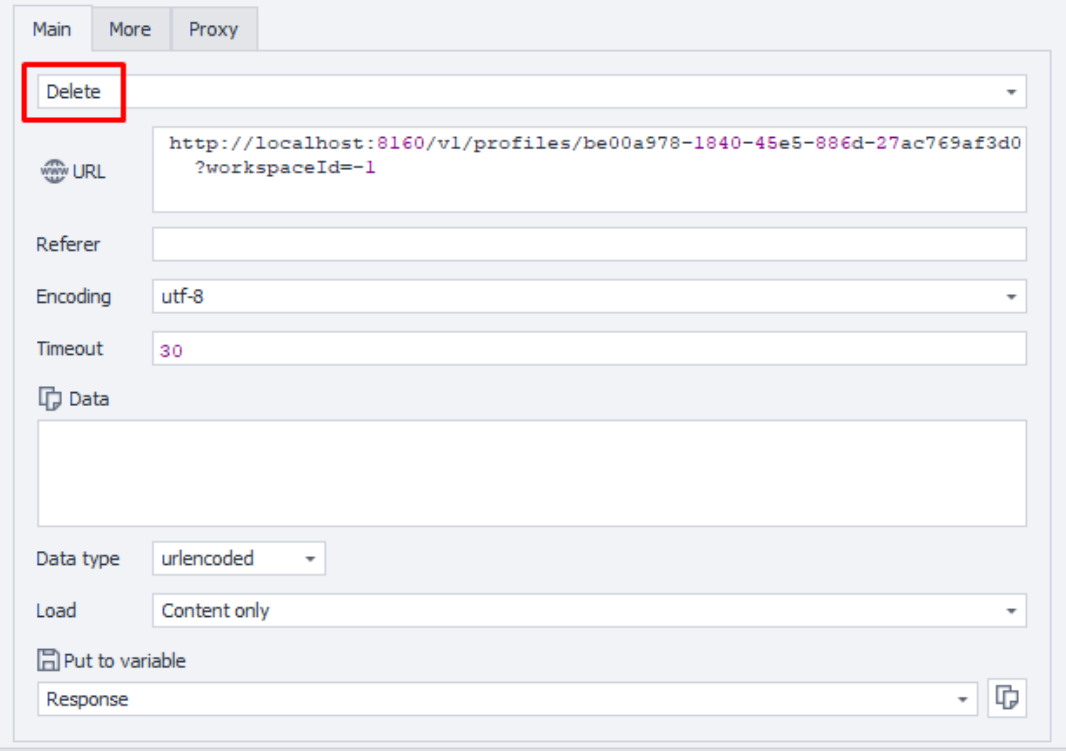
C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new Uri(
"http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
}
```

```
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

`http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1`



The screenshot shows the Postman application window with the 'Main' tab selected. A red rectangle highlights the 'Delete' method dropdown in the top left. The URL field contains the endpoint: `http://localhost:8160/v1/profiles/be00a978-1840-45e5-886d-27ac769af3d0?workspaceId=-1`. Other visible fields include 'Referer' (empty), 'Encoding' (set to 'utf-8'), 'Timeout' (set to '30'), 'Data' (empty), 'Data type' (set to 'urlencoded'), 'Load' (set to 'Content only'), and 'Put to variable' (set to 'Response').

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting browser profiles (bulk)

Description: This method is used to delete several existing browser profiles at once.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
body	JSON-Array	UUID profiles		Target Browser Profile IDs. Example: ["uuid1", "uuid2"]

Note:

The list of profiles which will be deleted should be specified as the array in the request body using the following format:

```
[
  "884d8f8d-9a4e-4b1e-9dd3-a028d3ae419b",
  "d24eba34-12d6-4412-ab20-80fae618c264"
]
```

Example request:

DELETE

CURL :

```
curl
'http://localhost:8160/v1/profiles/delete_bulk' \
--request DELETE \
--header 'Content-Type: application/json' \
--header 'Api-Token: YOUR_SECRET_TOKEN' \
--data '[
  "uuid1",
  "uuid2"
]'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new
Uri("http://localhost:8160/v1/profiles/delete_bulk"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
    Content = new StringContent("[\n  " +
"\n\"uuid1\", \" +
"\n\"uuid2\", \" +
"\n]\"")
    {
        Headers =
        {
            ContentType = new
MediaTypeHeaderValue("application/json")
        }
    }
}
```

```

    }
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

<http://localhost:8160/v1/profiles/{profileId}?workspaceId=-1>

Data:

`["uuid1", "uuid2"]`

The screenshot shows the Postman interface with the following configuration:

- Method:** Delete (highlighted with a red box)
- URL:** `http://localhost:8160/v1/profiles/delete_bulk`
- Referer:** (empty)
- Encoding:** utf-8
- Timeout:** 30
- Data:**

```
[
  "ae3228cc-fff3-4127-bd53-84054469eeb6",
  "7f549128-728f-4f8e-b8d1-5e0469e37a6a"
]
```
- Data type:** Other (dropdown) with `application/json` entered (highlighted with a red box)
- Load:** Content only
- Put to variable:** Response

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect
5

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Export cookies from a profile

Description: This method allows exporting cookies from an existing profile in JSON format.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
profileId	string	uuid	(required)	The unique identifier of the profile from which to export cookies (required) .

Example request:

POST

CURL:

```
curl 'http://localhost:8160/v1/profiles/{profileId}/cookies/export' \
  --request POST \
  --header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new
Uri("http://localhost:8160/v1/profiles/%7BprofileId%7D/cookies/expor
t"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

```
http://localhost:8160/v1/profiles/{profileId}/cookies/export
```

Main More Proxy

URL `http://localhost:8160/v1/profiles/{BF087C2F-F9A0-4458-828B-1A116E4836C6}/cookies/export`

Referer

Encoding `utf-8`

Timeout `30`

Data

Data type `urlencoded`

Load `Content only`

Put to variable

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main More Proxy

☒ Redirect `5`

☐ Use original url

Headers: `User settings`

Request headers:

`User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

Response API:

Response code	Result	Response
---------------	--------	----------

200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Import cookies into a profile

Description: This method allows importing cookies into an existing profile in JSON format.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
profileId	string	uuid	(required)	The unique identifier of the target profile for cookie import (required).
body	JSON-Array	UUID profiles		Cookies in JSON format [cookie]

Example request:

POST

CURL :

```
curl 'http://localhost:8160/v1/profiles/{profileId}/cookies/import' \
--request POST \
--header 'Content-Type: application/json' \
--header 'Api-Token: YOUR_SECRET_TOKEN' \
--data '{}'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new
Uri("http://localhost:8160/v1/profiles/%7BprofileId%7D/cookies/import"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
    Content = new StringContent("{}")
    {
        Headers =
        {
            ContentType = new
MediaTypeHeaderValue("application/json")
        }
    }
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/profiles/{profileId}/cookies/import
```

Main More Proxy

URL `http://localhost:8160/v1/profiles/{F1775453-46A2-4E51-BC1A-20D813E50979}/cookies/import`

Referer

Encoding `utf-8`

Timeout `30`

Data

```
[{"name": "_ga", "value": "GA1.1.1062752758.1764746459", "domain": ".pixelscan.net", "path": "/", "expirationDate": 1799306458.742868, "secure": false, "httpOnly": false, "sameSite": "unspecified", "sourceScheme": "secure", "sourcePort": 443, "sourceType": "script", "hostOnly": false, "session": false}, {"name": "_ga_ZH1CYL80NM", "value": "GS2.1.s1764746458%ol$gl$tl764746501$j17$10$h0", "domain": ".pixelscan.net", "path": "/", "expirationDate": 1799306501.983707, "secure": false, "httpOnly": false, "sameSite": "unspecified", "sourceScheme": "secure", "sourcePort": 443, "sourceType": "script", "hostOnly": false, "session": false}]
```

Data type `Other` `application/json`

Load `Content only`

Put to variable

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main More Proxy

☒ Redirect `s`

☐ Use original url

Headers: `User settings`

Request headers:

```
User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}
```

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Proxy Management

Getting proxy list with parameters

Description: This method provides functionality for filtering, sorting, and paginating proxies. It is used to utilize the available parameters to customize the output. If parameters are not specified, the response will return the default page with results in standard order.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
start	integer	int32	0	Starting index for the list of proxies to retrieve.
total	integer	int32	1000	Maximum number of proxies to retrieve.
folderId	string	uuid	(empty)	Optional folder ID for filtering proxies. Pass null to ignore this filter.
id	string	uuid	(empty)	Optional proxy ID to filter. Pass null to ignore this filter.
name	string		(empty)	Optional name for filtering proxies. Pass null to ignore this filter.
sorting	string		(empty)	Optional sorting string to determine the order of the returned proxies. Pass null to ignore this filter.

Example request:

GET

CURL :

```
curl
'http://localhost:8160/v1/proxies?workspaceId=&start=&total=&folderId=&id=&name=&sorting=' \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
```

```

var request = new HttpRequestMessage
{
    Method = HttpMethod.Get,
    RequestUri = new Uri(
"http://localhost:8160/v1/proxies?workspaceId=&start=&total=&folderId=&id=&name=&sorting="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

<http://localhost:8160/v1/proxies?workspaceId=&start=&total=&folderId=&id=&name=&sorting=>

The screenshot shows a web client interface with the following fields and values:

- URL:** `http://localhost:8160/v1/proxies?workspaceId=&start=&total=&folderId=&id=&name=&sorting=`
- Referer:** (empty)
- Encoding:** `utf-8`
- Timeout:** `30`
- Load:** `Content only`
- Put to variable:** `Listlistbad`

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre> { "totalCount": 1, "items": [{ "id": "123e4567-e89b-12d3-a456-426614174000", "folderId": "123e4567-e89b-12d3-a456-426614174000", "name": null, "proxyUri": null, "checkStatus": "Unknown", "checkStatusLastUpdateTime": null, "ipChangeUrl": null }] } </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Adding new proxy

Description: This method is used to add a new proxy entry using the provided configuration parameters. It returns the generated unique identifier.

Request parameters:

Parameter	Type	Format	Default	Description
name	string		(empty)	Name of the added proxy. This value cannot be null or empty.
proxyUrl	string		(empty)	URL-address of the proxy. This value must be a valid URL. (cannot be null or empty, for example http://login:pass@host:port).
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
folderId	string	uuid	(empty)	Optional identifier of the folder to associate with the proxy. Can take null value.
ipChangeUrl	string		(empty)	Optional URL to trigger IP change for the proxy. Defaults to an empty string if not specified.

Example request:

POST

CURL:

```
curl
'http://localhost:8160/v1/proxies/create?name=&proxyUri=&workspaceId=&f
olderId=&ipChangeUrl=' \
--request POST \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new
Uri("http://localhost:8160/v1/proxies/create?name=&proxyUri=&workspaceI
d=&folderId=&ipChangeUrl="),
```

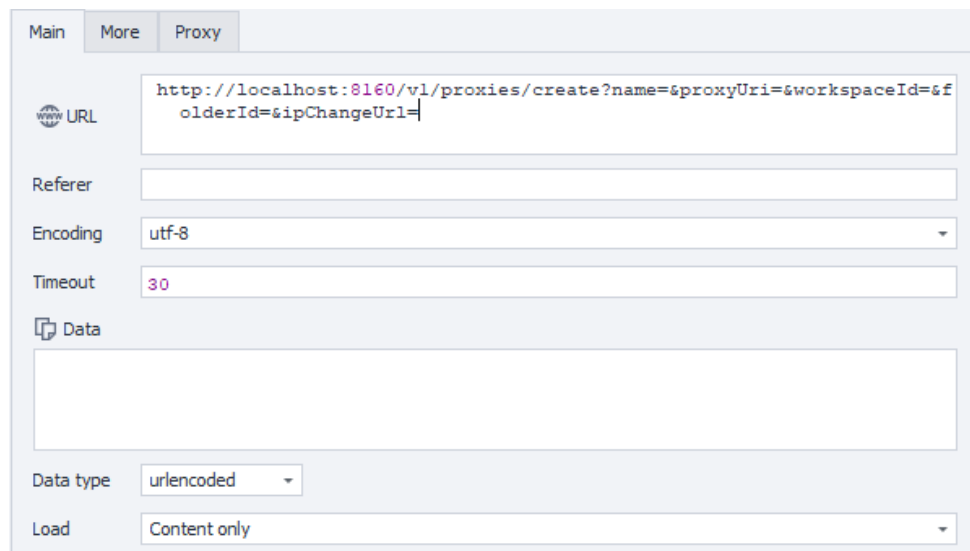
```

    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

<http://localhost:8160/v1/proxies/create?name=&proxyUri=&workspaceId=&folderId=&ipChangeUrl=>



The screenshot shows the Apache JMeter GUI with the 'Proxy' tab selected. The 'URL' field contains the URL: `http://localhost:8160/v1/proxies/create?name=&proxyUri=&workspaceId=&folderId=&ipChangeUrl=`. The 'Referer' field is empty. The 'Encoding' dropdown is set to 'utf-8'. The 'Timeout' field is set to '30'. The 'Data' section is empty. The 'Data type' dropdown is set to 'urlencoded'. The 'Load' dropdown is set to 'Content only'.

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre>id of the newly created proxy ["123e4567-e89b-12d3-a456-426614174000"]</pre>
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Editing existing proxy

Description: This method modifies configuration of an existing proxy server based on specified parameters.

Request parameters:

Parameter	Type	Format	Default	Description
proxyId	string	uuid	(empty)	Unique identifier of the proxy (Required)

workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
folderId	string	uuid	(empty)	New folder for the proxy. Set to empty to leave the current value unchanged.
name	string		(empty)	New name for the proxy. Set to null to retain the current name.
proxyUrl	string		(empty)	New URL for the proxy.
ipChangeUrl	string		(empty)	New URL for IP rotation.

Note: The ID of the edited proxy should be specified inside curly braces { } in the URL path.

Example request:

PUT

CURL:

```
curl
'http://localhost:8160/v1/proxies/{proxyId}?workspaceId=&folderId=&name=
&proxyUri=&ipChangeUrl=' \
--request PUT \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Put,
    RequestUri = new
Uri("http://localhost:8160/v1/proxies/{proxyId}?workspaceId=&folderId=&
name=&proxyUri=&ipChangeUrl="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
}
```

```
Console.WriteLine(body);
}
```

Cube:

<http://localhost:8160/v1/proxies/{proxyId}?workspaceId=&folderId=&name=&proxyUri=&ipChangeUrl=>

The screenshot shows the 'Put' method configuration in a REST client. The 'URL' field is highlighted with a red box and contains the endpoint: `http://localhost:8160/v1/proxies/{proxyId}?workspaceId=&folderId=&name=&proxyUri=&ipChangeUrl=`. Below the URL field, there are fields for 'Referer', 'Encoding' (set to 'utf-8'), 'Timeout' (set to '30'), and 'Data'. The 'Data' field is empty. At the bottom, 'Data type' is set to 'urlencoded' and 'Load' is set to 'Content only'.

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

The screenshot shows the 'More' tab in the REST client. The 'Redirect' checkbox is checked and set to '5'. The 'Use original url' checkbox is unchecked. The 'Headers' dropdown is set to 'User settings'. The 'Request headers' field contains the following headers: `User-Agent: {-Profile.UserAgent-}` and `Api-Token: {-Variable.Token-}`. The 'Use Cookie Container' checkbox is checked.

Response API:

Response code	Result	Response
---------------	--------	----------

200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting proxy

Description: This method deletes proxy from a list by specified ID.

Request parameters:

Parameter	Type	Format	Default	Description
proxyId	string	uuid	(empty)	Unique identifier of the proxy (required).
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note: The ID of the edited proxy should be specified inside curly braces `{}` in the URL path.

Example request:

DELETE

CURL :

```
curl 'http://localhost:8160/v1/proxies/{proxyId}?workspaceId=' \
  --request DELETE \
  --header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new
Uri("http://localhost:8160/v1/proxies/{proxyId}?workspaceId="),
    Headers =
```

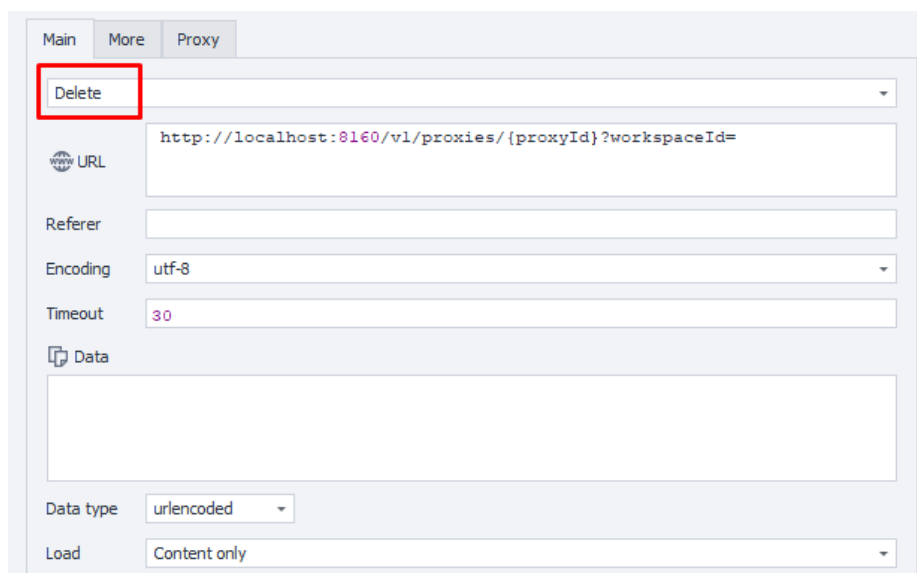
```

{
    { "Api-Token", "YOUR_SECRET_TOKEN" },
},
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

<http://localhost:8160/v1/proxies/{proxyId}?workspaceId=>



The screenshot shows the Burp Suite Proxy tab with the following configuration:

- Method:** Delete (highlighted with a red box)
- URL:** `http://localhost:8160/v1/proxies/{proxyId}?workspaceId=`
- Referer:** (empty)
- Encoding:** utf-8
- Timeout:** 30
- Data:** (empty text area)
- Data type:** urlencoded
- Load:** Content only

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting proxies (bulk)

Description: This method sends a bulk delete request to the API to remove specified proxies.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
body	JSON array	GUID[]		The array of UUIDs containing the proxies to be deleted.

Note:

The list of proxies to delete should be specified as the array in the request body using the following format:

```
[
  "884d8f8d-9a4e-4b1e-9dd3-a028d3ae419b",
  "d24eba34-12d6-4412-ab20-80fae618c264"
]
```

Example request:

DELETE

CURL:

```
curl 'http://localhost:8160/v1/proxies/delete_bulk?workspaceId=' \
--request DELETE \
--header 'Content-Type: application/json' \
--header 'Api-Token: YOUR_SECRET_TOKEN' \
--data '[
  ""
]'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new
Uri("http://localhost:8160/v1/proxies/delete_bulk?workspaceId="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
    Content = new StringContent("[\n \"\"\n]")
    {
        Headers =
        {
            ContentType = new MediaTypeHeaderValue("application/json")
        }
    }
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

}

Cube:

`http://localhost:8160/v1/proxies/delete_bulk?workspaceId=`

The screenshot shows the 'Main' tab of a web client interface. The 'Method' dropdown is set to 'Delete'. The 'URL' field contains the endpoint `http://localhost:8160/v1/proxies/delete_bulk?workspaceId=`. The 'Referer' field is empty. The 'Encoding' is set to 'utf-8' and the 'Timeout' is '30'. The 'Data' section shows a JSON array: `[{"-Variable.Proxy2-"}, {"-Variable.Proxy3-"}]`. The 'Data type' is set to 'Other' with a value of 'application/json'. The 'Load' option is set to 'Content only'.

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

The screenshot shows the 'More' tab of the web client interface. The 'Redirect' checkbox is checked with a value of '5'. The 'Use original url' checkbox is unchecked. The 'Headers' dropdown is set to 'User settings'. The 'Request headers' section contains two entries: 'User-Agent: `{-Profile.UserAgent-}`' and 'Api-Token: `{-Variable.Token-}`'. The 'Use Cookie Container' checkbox is checked.

Response API:

Response code	Result	Response
200 OK	Success	

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------------	--

Proxy Folders Management

Getting proxy folder list

Description: This method is used to get the list of proxy folders. It supports filtering, sorting, and pagination in returned results. Optional query parameters can be used to refine the result set. If parameters are not specified, the method returns the first page of proxy folders in the default sorting order.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
start	integer	int32	0	Index of the first folder to return (zero-based)
total	integer	int32	1000	Maximum number of results to return
id	string	uuid	(empty)	Filter by certain folder identifier
name	string		(empty)	Filter by folder name
sorting	string		(empty)	Result sorting condition
location	string (enum)		(empty)	Proxy type filter (Local or Cloud)

Example request:

GET

CURL :

```
curl
'http://localhost:8160/v1/proxy_folders?workspaceId=&start=&total=&id=&name=&sorting=&=' \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
```

```

        Method = HttpMethod.Get,
        RequestUri = new
Uri("http://localhost:8160/v1/proxy_folders?workspaceId=&start=&total=&
id=&name=&sorting=&="),
        Headers =
        {
            { "Api-Token", "YOUR_SECRET_TOKEN" },
        },
    };
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

http://localhost:8160/v1/proxy_folders?workspaceId=&start=&total=&id=&name=&sorting=&=

The screenshot shows the Proxyman application interface with the 'Proxy' tab selected. The configuration fields are as follows:

- URL:** http://localhost:8160/v1/proxy_folders?workspaceId=&start=&total=&id=&name=&sorting=&=
- Referer:** (empty field)
- Encoding:** utf-8
- Timeout:** 30
- Load:** Content only
- Put to variable:** Listlistbad

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre> { "totalCount": 1, "items": [{ "id": "123e4567-e89b-12d3-a456-426614174000", "name": null, "location": "Local", "quantity": 1 }] } </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Creating proxy folder

Description: This method creates a new proxy folder.

Request parameters:

Parameter	Type	Format	Default	Description
-----------	------	--------	---------	-------------

name	string		(empty)	The name of the new folder (Required)
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
location	string (enum)		Local	Storage location (Local or Cloud)

Example request:

POST

CURL :

```
curl 'http://localhost:8160/v1/proxy_folders?name=&workspaceId=&=' \
  --request POST \
  --header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new
Uri("http://localhost:8160/v1/proxy_folders?name=&workspaceId=&="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

http://localhost:8160/v1/proxy_folders?name=&workspaceId=&=

Main
More
Proxy

 URL

http://localhost:8160/v1/proxy_folders?name=&workspaceId=&=

Referer

Encoding
utf-8

Timeout
30

 Data

Data type
urlencoded

Load
Content only

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect
s

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	The identifier of the newly created proxy list. (123e4567-e89b-12d3-a456-426614174000)

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------	----------------------------------

Editing proxy folder

Description: This method updates the name of the specified proxy folder. This is the only editable property.

Request parameters:

Parameter	Type	Format	Default	Description
proxyFolderId	string	uuid	(empty)	The folder identifier (Required)
name	string		(empty)	The new folder name
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note:

The ID of the edited proxy list should be specified inside curly braces {} in the URL path.

Example request:

PUT

CURL:

```
curl
'http://localhost:8160/v1/proxy_folders/{proxyFolderId}?name=&workspace
Id=' \
--request PUT \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Put,
```

```

    RequestUri = new
Uri("http://localhost:8160/v1/proxy_folders/{proxyFolderId}?name=&workspaceId="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

http://localhost:8160/v1/proxy_folders/{proxyFolderId}?name=&workspaceId=

The screenshot shows a web client interface with tabs for 'Main', 'More', and 'Proxy'. The 'Main' tab is active. A red box highlights the 'Put' method dropdown. The URL field contains 'http://localhost:8160/v1/proxy_folders/{proxyFolderId}?name=&workspaceId='. Other fields include 'Referer' (empty), 'Encoding' (set to 'utf-8'), 'Timeout' (set to '30'), 'Data' (empty text area), 'Data type' (set to 'urlencoded'), and 'Load' (set to 'Content only').

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting proxy folder

Description: This method deletes a proxy folder. The parameter `proxyFolderId` must match the existing folder within the specified workspace.

Request parameters:

Parameter	Type	Format	Default	Description
<code>proxyFolderId</code>	string	uuid	(empty)	The identifier of the proxy folder to be deleted (Required)
<code>workspaceId</code>	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note:

The ID of the deleted proxy folder should be specified inside curly braces {} in the URL path.

Example request:**DELETE****CURL :**

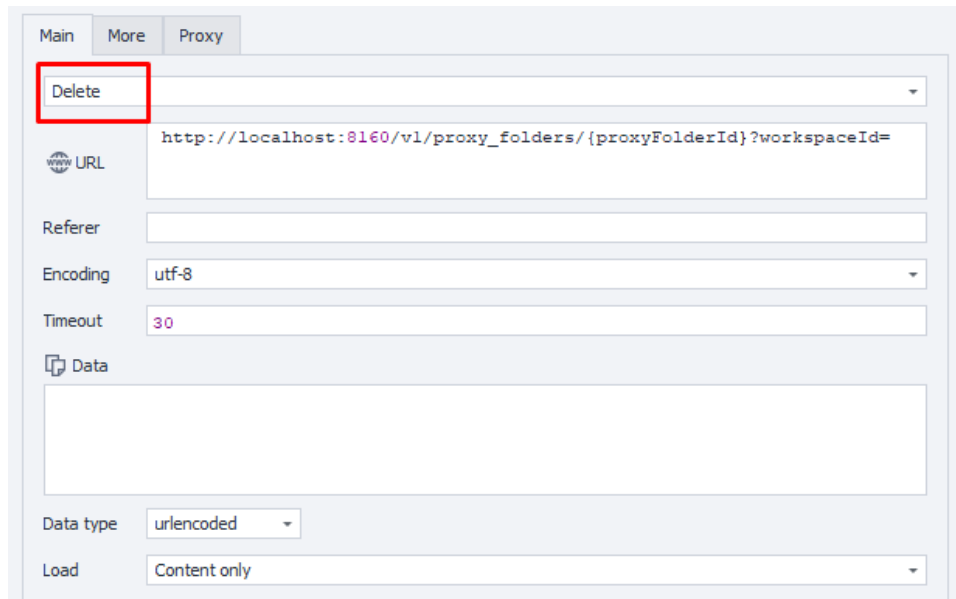
```
curl
'http://localhost:8160/v1/proxy_folders/{proxyFolderId}?workspaceId=' \
--request DELETE \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new
Uri("http://localhost:8160/v1/proxy_folders/{proxyFolderId}?workspaceId
="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

http://localhost:8160/v1/proxy_folders/{proxyFolderId}?workspaceId=



Main More Proxy

Delete

URL `http://localhost:8160/v1/proxy_folders/{proxyFolderId}?workspaceId=`

Referer

Encoding `utf-8`

Timeout `30`

Data

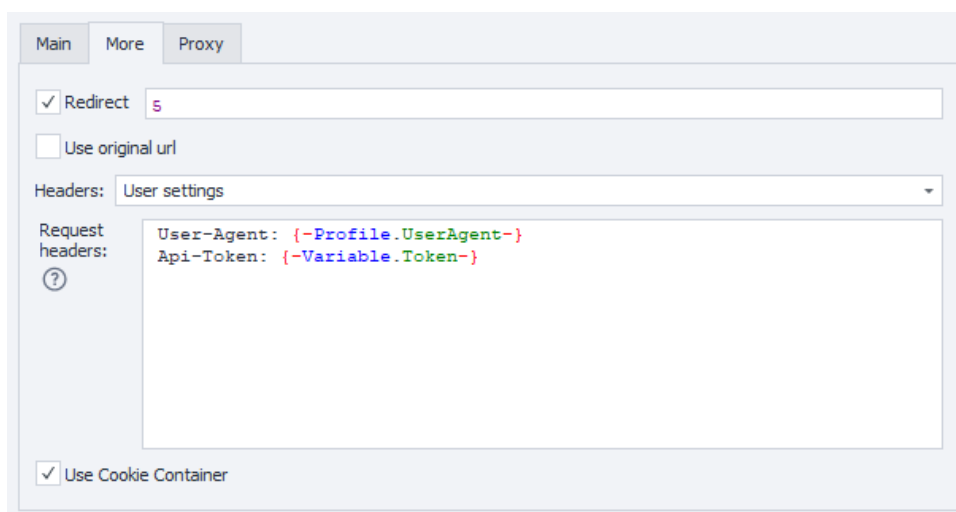
Data type `urlencoded`

Load `Content only`

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.



Main More Proxy

☒ Redirect `s`

☐ Use original url

Headers: `User settings`

Request headers: `User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

 Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Browser Instances

Getting browser instances

Description: This method returns the list of browser processes available for the current user with detailed information.

Results can be filtered by profile ID or process ID and optionally sorted or limited by number of returned entries.

Important: These actions are applied **only** to browser instances launched via the API.

Request parameters:

Parameter	Type	Format	Default	Description
<code>workspaceId</code>	integer	int64	-1	Workspace identifier. <code>-1</code> means the default workspace.
<code>start</code>	integer	int32	0	Starting index for pagination
<code>total</code>	integer	int32	1000	Maximum number of results to return
<code>profileId</code>	string	uuid	(empty)	Filter by profile ID (optional).
<code>processId</code>	integer	int32	(empty)	Filter by browser process ID (optional).
<code>sorting</code>	string		(empty)	Sorting parameters: <code>ConnectionString</code> , <code>ProcessId</code> , <code>ProfileId</code> (optional).

Example request:

GET

CURL :

```
curl
'http://localhost:8160/v1/browser_instances?workspaceId=-1&start=0&total=1000&profileId=&processId=&sorting=' \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
```

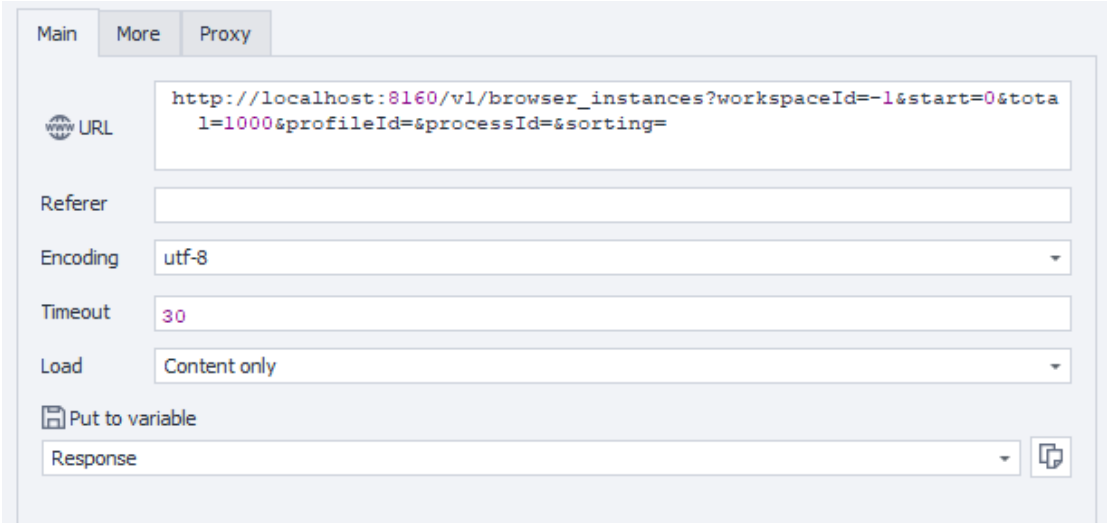
```

var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Get,
    RequestUri = new Uri(
"http://localhost:8160/v1/browser_instances?workspaceId=-1&start=0&total=1000&profileId=&processId=&sorting="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

http://localhost:8160/v1/browser_instances?workspaceId=-1&start=0&total=1000&profileId=&processId=&sorting=



The screenshot shows a web client interface with the following fields and values:

- URL:** http://localhost:8160/v1/browser_instances?workspaceId=-1&start=0&total=1000&profileId=&processId=&sorting=
- Referer:** (empty field)
- Encoding:** utf-8
- Timeout:** 30
- Load:** Content only
- Put to variable:** Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre> { "totalCount": 1, "items": [{ "profileId": "123e4567-e89b-12d3-a456-426614174000", "processId": 1, "connectionString": null }] }</pre>
500 Error	Internal Server Error	<pre> { "message": null }</pre>

Creating browser instance

Description: This method creates the new browser instance using the specified profile with optional workspace and desktop settings.

Request parameters:

Parameter	Type	Format	Default	Description
profileId	string	uuid	(required)	Unique identifier of the profile to associate with the browser instance.
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
desktopName	string		(empty)	The name of the desktop environment (optional).
threadToken	string		(empty)	Thread token for the browser instance. To create a new execution thread (optional) pass empty string (default).

Example request:

POST

CURL:

```
curl
'http://localhost:8160/v1/browser_instances/create?profileId=PROFILE_ID&workspaceId=-1&desktopName=&threadToken=' \
--request POST \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri(
"http://localhost:8160/v1/browser_instances/create?profileId=PROFILE_ID&workspaceId=-1&desktopName=&threadToken="),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
}
```

```

var body = await response.Content.ReadAsStringAsync();
Console.WriteLine(body);
}

```

Cube:

http://localhost:8160/v1/browser_instances/create?profileId=PROFILE_ID&workspaceId=-1&desktopName=&threadToken=

The screenshot shows the 'Main' tab of a web client interface. The URL field contains the request: `http://localhost:8160/v1/browser_instances/create?profileId=EF25ED3A-4A37-4E5C-AB5B-7C5B8AF1058B&workspaceId=-1&desktopName=&threadToken=`. The Referer field is empty. The Encoding is set to 'utf-8'. The Timeout is set to '30'. The Data type is set to 'urlencoded'. The Load is set to 'Content only'. The Put to variable dropdown is set to 'Response'.

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

The screenshot shows the 'More' tab of the web client interface. The Redirect checkbox is checked with a value of '5'. The Use original url checkbox is unchecked. The Headers dropdown is set to 'User settings'. The Request headers field contains: `User-Agent: {-Profile.UserAgent-}` and `Api-Token: {-Variable.Token-}`. The Use Cookie Container checkbox is checked.

Response API:

Response code	Result	Response
200 OK	Success	<pre>{ "profileId": "123e4567-e89b-12d3-a456-426614174000", "processId": 1, "connectionString": null }</pre>
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Creating multiple browser instances

Description: This method is used to create browser instances in bulk for the specified profiles. Please ensure that the provided profile IDs are valid and the specified workspace ID exists.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
desktopName	string		(empty)	The name of the desktop environment (optional).
keepAlive	логическое значение		true	This value indicates if the browser instances should remain active after creation. Set "true" to keep the instances active; otherwise, false. Defaults to true (true или false).
body	JSON-Array			The array of profile UUIDs to create browser instances for. Example: ["uuid1", "uuid2"]

Note:

The list of browser instances to create should be passed as the array in the request body using the following format:

```
[  
  "884d8f8d-9a4e-4b1e-9dd3-a028d3ae419b",  
  "d24eba34-12d6-4412-ab20-80fae618c264"  
]
```

Example request:**POST****CURL:**

```
curl  
'http://localhost:8160/v1/browser_instances/create_bulk?workspaceId=  
-1&desktopName=&keepAlive=true' \  
--request POST \  
--header 'Content-Type: application/json' \  
--header 'Api-Token: YOUR_SECRET_TOKEN' \  
--data '[  
  "uuid1",  
  "uuid2"  
]'
```

C#:

```
using System.Net.Http.Headers;  
var client = new HttpClient();  
var request = new HttpRequestMessage  
{  
    Method = HttpMethod.Post,  
    RequestUri = new Uri(  
"http://localhost:8160/v1/browser_instances/create_bulk?workspaceId=  
-1&desktopName=&keepAlive=true"),  
    Headers =  
    {  
        { "Api-Token", "YOUR_SECRET_TOKEN" },  
    },  
    Content = new StringContent("[\n  " +  
"\n\"uuid1\", \" +  
"\n\"uuid2\", \" +  
"\n]")
```

```

{
    Headers =
    {
        ContentType = new
MediaTypeHeaderValue("application/json")
    }
}
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

```

Cube:

http://localhost:8160/v1/browser_instances/create_bulk?workspaceId=-1&desktopName=&keepAlive=true

Data:

`["uuid1", "uuid2"]`

The screenshot shows a web client interface with the following fields:

- URL:** `http://localhost:8160/v1/browser_instances/create_bulk?workspaceId=-1&desktopName=&keepAlive=true`
- Referer:** (empty)
- Encoding:** `utf-8`
- Timeout:** `30`
- Data:**

```
[
  "d5f3a440-d70f-4216-895b-54f108bc2508",
  "c98b47b8-568e-41e1-bc2c-d9e015488d0b"
]
```
- Data type:** `Other` (dropdown menu) with `application/json` entered in the adjacent text field.
- Load:** `Content only` (dropdown menu)
- Put to variable:** `Response` (dropdown menu)

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	<pre>[{ "profileId": "123e4567-e89b-12d3-a456-426614174000", "processId": 1, "connectionString": null }]</pre>
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting browser instance

Description: This method deletes the existing browser instance associated with the specified profile.

This operation permanently closes the browser instance and releases all associated resources.

Request parameters:

Parameter	Type	Format	Default	Description
-----------	------	--------	---------	-------------

profileId	string	uuid	(required)	The target browser instance ID
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Example request:

DELETE

CURL:

```
curl
'http://localhost:8160/v1/browser_instances/{profileId}?workspaceId=-1' \
  --request DELETE \
  --header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new Uri(
"http://localhost:8160/v1/browser_instances/{profileId}?workspaceId=-1"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/browser_instances/{profileId}?workspaceId=-1
```

Main
More
Proxy

Delete

URL
http://localhost:8160/v1/browser_instances/d5f3a440-d70f-4216-895b-54f108bc2508?workspaceId=-1

Referer

Encoding
utf-8

Timeout
30

Data

Data type
urlencoded

Load
Content only

Put to variable
Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main
More
Proxy

☒ Redirect 5

☐ Use original url

Headers: User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------	----------------------------------

Deleting browser instances (bulk)

Description: This method deletes multiple existing browser instances associated with the specified profiles.

This operation permanently closes all specified browser instances and releases their resources.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
body	JSON-Array			The array of profile UUIDs of those browser instances which should be deleted. For example: ["uuid1", "uuid2"]

Note:

The list of browser instances to delete should be specified as the array in the request body using the following format:

```
[
  "884d8f8d-9a4e-4b1e-9dd3-a028d3ae419b",
  "d24eba34-12d6-4412-ab20-80fae618c264"
]
```

Example request:

DELETE

CURL :

```
curl
'http://localhost:8160/v1/browser_instances/delete_bulk?workspaceId=-1' \
--request DELETE \
--header 'Content-Type: application/json' \
```

```
--header 'Api-Token: YOUR_SECRET_TOKEN' \
--data '[
  "uuid1",
  "uuid2"
]'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Delete,
    RequestUri = new Uri(
"http://localhost:8160/v1/browser_instances/delete_bulk?workspaceId=-1"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
    Content = new StringContent("[\n  " +
"\n\"uuid1\", \" +
"\n\"uuid2\", \" +
"\n]\n")
    {
        Headers =
        {
            ContentType = new
MediaTypeHeaderValue("application/json")
        }
    }
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Cube:

http://localhost:8160/v1/browser_instances/delete_bulk?workspaceId=-1

Data:

`["uuid1", "uuid2"]`

The screenshot shows a REST client interface with the following configuration:

- Method:** Delete (highlighted with a red box)
- URL:** `http://localhost:8160/v1/browser_instances/delete_bulk?workspaceId=-1`
- Referer:** (empty)
- Encoding:** utf-8
- Timeout:** 30
- Data:**

```
[  
  "c98b47b8-568e-41e1-bc2c-d9e015488d0b",  
  "a68b47b8-568e-41e1-bc2c-d9e015488d19",  
]
```
- Data type:** Other (dropdown) and `application/json` (text input, highlighted with a red box)
- Load:** Content only
- Put to variable:** Response

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

The screenshot shows a REST client interface with the following configuration:

- Redirect:** ☒ Redirect 5
- Use original url:** ☐
- Headers:** User settings
- Request headers:**

```
User-Agent: {-Profile.UserAgent-}  
Api-Token: {-Variable.Token-}
```
- Use Cookie Container:** ☒

Response API:

Response code	Result	Response
200 OK	Success	

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------------	--

Thread Management

Getting execution threads

Description: This method retrieves the list of execution threads for the specified workspace. Threads can be filtered by token or type, limited by number of results, and sorted optionally.

Request parameters:

Parameter	Type	Format	Default	Description
<code>workspaceId</code>	integer	int64	-1	Workspace identifier. <code>-1</code> means the default workspace.
<code>start</code>	integer	int32	0	Starting index for pagination.
<code>total</code>	integer	int32	1000	Maximum number of threads to return.
<code>thread_token</code>	string		(empty)	Filter by certain thread token (optional).
<code>type</code>	string enum		(empty)	Filter by token type (<code>Manual</code> , <code>BrowserInstance</code>) (optional).
<code>sorting</code>	string		(empty)	Sorting parameters: <code>CreatedAt</code> , <code>ThreadToken</code> , <code>Type</code> , <code>WorkspaceId</code> (optional).

Example request:

GET

CURL :

```
curl
```

```
'http://localhost:8160/v1/threads?workspaceId=&start=0&total=1000&th  
read_token=&type=&sorting=' \n  
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;  
var client = new HttpClient();  
var request = new HttpRequestMessage  
{  
    Method = HttpMethod.Get,  
    RequestUri = new Uri(  
"http://localhost:8160/v1/threads?workspaceId=&start=0&total=1000&th  
read_token=&type=&sorting="),  
    Headers =  
    {  
        { "Api-Token", "YOUR_SECRET_TOKEN" },  
    },  
};  
using (var response = await client.SendAsync(request))  
{  
    response.EnsureSuccessStatusCode();  
    var body = await response.Content.ReadAsStringAsync();  
    Console.WriteLine(body);  
}
```

Cube:

http://localhost:8160/v1/threads?workspaceId=&start=&total=&thread_token=&&sorting=

The screenshot shows the 'Main' tab of a web proxy tool. The 'URL' field contains the request: `http://localhost:8160/v1/threads?workspaceId=&start=0&total=1000&thread_token=&type=&sorting=`. The 'Referer' field is empty. The 'Encoding' dropdown is set to 'utf-8'. The 'Timeout' field is set to '30'. The 'Load' dropdown is set to 'Content only'. At the bottom, the 'Put to variable' section has a dropdown set to 'Response' and a copy icon.

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

The screenshot shows a configuration window with three tabs: 'Main', 'More', and 'Proxy'. The 'Main' tab is active. It contains a 'Redirect' checkbox checked with a value of '5', a 'Use original url' checkbox unchecked, a 'Headers' dropdown menu set to 'User settings', and a 'Request headers' section with a text area containing:
User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}
At the bottom, there is a 'Use Cookie Container' checkbox checked.

Response API:

Response code	Result	Response
200 OK	Success	<pre>{ "totalCount": 1, "items": [{ "workspaceId": 1, "threadToken": null, "type": "Manual", "createdAt": "2025-07-25T09:34:29.768Z" }] }</pre>
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Running thread**Description:** This method runs a new execution thread.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Example request:

POST

CURL:

```
curl
'http://localhost:8160/v1/threads/create?workspaceId=-1' \
--request POST \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri(
"http://localhost:8160/v1/threads/create?workspaceId=-1"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/threads/create?workspaceId=-1
```

Main More Proxy

URL `http://localhost:8160/v1/threads/create?workspaceId=-1`

Referer

Encoding `utf-8`

Timeout `30`

Data

Data type `urlencoded`

Load `Content only`

Put to variable `Response`

Additionally:

User-Agent: `{-Profile.UserAgent-}`

Api-Token: Token from **UserArea2**.

Main More Proxy

☒ Redirect `5`

☐ Use original url

Headers: `User settings`

Request headers: `User-Agent: {-Profile.UserAgent-}`
`Api-Token: {-Variable.Token-}`

☒ Use Cookie Container

Response API:

Response code	Result	Response
<code>200 OK</code>	<code>Success</code>	<code>{</code> <code> "workspaceId": 1,</code>

		<pre> "threadToken": null, "type": "Manual", "createdAt": "2025-07-25T09:34:29.768Z" } </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Running multiple threads

Description: This method can be used to start execution threads in bulk.

Request parameters:

Parameter	Type	Format	Default	Description
count	integer	int32		Number of threads to run.
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note:

The number of threads to run should be specified as a path parameter in the URL.

Example request:

POST

CURL :

```

curl
'http://localhost:8160/v1/threads/create_bulk?count=5&workspaceId=-1' \
--request POST \
--header 'Api-Token: YOUR_SECRET_TOKEN'

```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new Uri("
http://localhost:8160/v1/threads/create_bulk?count=5&workspaceId=-1"
),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/threads/create_bulk?count=5&workspaceId=-1
```

MainMoreProxy

 URL

http://localhost:8160/v1/threads/create_bulk?count=5&workspaceId=-1

Referer

Encoding

utf-8

Timeout

30

 Data

Data type

urlencoded

Load

Content only

 Put to variable

Response

Additionally:
User-Agent: {-Profile.UserAgent-}
Api-Token: Token from **UserArea2**.

MainMoreProxy

☒ Redirect

s

☐ Use original url

Headers:

User settings

Request headers:

User-Agent: {-Profile.UserAgent-}

Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	[{

		<pre> "workspaceId": 1, "threadToken": null, "type": "Manual", "createdAt": "2025-07-25T09:34:29.768Z" }] </pre>
500 Error	Internal Server Error	<pre> { "message": null } </pre>

Updating execution thread

Description: This method is used to continue work of the execution thread associated with the thread token.

Request parameters:

Parameter	Type	Format	Default	Description
threadToken	string		(required)	Unique identifier of the execution thread to continue.
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note:

The ID of the updated thread should be specified inside curly braces {} in the URL path.

Example request:

PUT

CURL:

```

curl
'http://localhost:8160/v1/threads/use/{threadToken}?workspaceId=-1'
\
--request PUT \

```

```
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Put,
    RequestUri = new Uri(
"http://localhost:8160/v1/threads/use/{threadToken}?workspaceId=-1")
,
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

Curl:

```
http://localhost:8160/v1/threads/use/{threadToken}?workspaceId=-1
```

Main More Proxy

Put

URL <http://localhost:8160/v1/threads/use/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZWlncy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnp2mllicI6ImI3YjE3ZTZjLWI3MGYtNGZiOS1...>

Referer

Encoding utf-8

Timeout 30

Data

Data type urlencoded

Load Content only

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main More Proxy

☒ Redirect 5

☐ Use original url

Headers: User settings

Request headers:

User-Agent: {-Profile.UserAgent-}

Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	

500 Error	Internal Server Error	<pre>{ "message": null }</pre>
-----------	-----------------------	----------------------------------

Updating multiple threads

Description: This method is used to update the specified execution threads allowing them to continue working. The threadTokens parameter should contain the array of valid tokens with existing workspace ID's.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
body	JSON-Array			The array of thread tokens to update execution threads. Cannot be null or empty. Example: ["uuid1", "uuid2"]

Note:

The list of threads to update should be specified as the array in the request body using the following format:

```
[
  "threadToken1",
  "threadToken2"
]
```

Example request:

PUT

CURL:

```
curl
'http://localhost:8160/v1/threads/use_bulk?workspaceId=-1' \
--request PUT \
--header 'Content-Type: application/json' \
--header 'Api-Token: YOUR_SECRET_TOKEN' \
--data '[
    "threadToken1",
    "threadToken2"
]'
```

C#:

```
using System.Net.Http.Headers;
var client = new HttpClient();
var request = new HttpRequestMessage
{
    Method = HttpMethod.Put,
    RequestUri = new Uri(
"http://localhost:8160/v1/threads/use_bulk?workspaceId=-1"),
    Headers =
    {
        { "Api-Token", "YOUR_SECRET_TOKEN" },
    },
    Content = new StringContent("[\n " +
"\nthreadToken1\n", " +
"\nthreadToken2\n", " +
"\n]")
};

{
    Headers =
    {
        ContentType = new
MediaTypeHeaderValue("application/json")
    }
}

};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}
```

`http://localhost:8160/v1/threads/use_bulk?workspaceId=-1`

Data :

```
["threadToken1", "threadToken2"]
```

Main

More

Proxy

Put

URL

http://localhost:8160/v1/threads/use_bulk?workspaceId=-1

Referer

Encoding

utf-8

Timeout

30

Data

[
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZWlkecyS4bWxzb2FwLm9yZy93cy8yMDAxLzA1L2lkZW50aXR5L2NsYWltcy9uYWI1aWR1bnRpZmllciI6ImI3YjE3ZTZjLWl3MGYtNGZiOS1hNzNjLWNjZWZjOWM3NDdiZSI6Imh0dHA6Ly9zY2hlbWFzLnhtbHNvYXAub3JnL3dzL"

Data type

Other

application/json

Load

Content only

Put to variable

Response

Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

Main

More

Proxy

☒ Redirect

☐ Use original url

Headers:

User settings

Request headers:

?

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response	Result	Response
----------	--------	----------

code		
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting execution thread

Description: This method can be used to delete an existing execution thread.

Request parameters:

Parameter	Type	Format	Default	Description
threadToken	string		(required)	Target thread token (required).
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.

Note:

The ID of the thread to delete should be specified inside curly braces {} in the URL path.

Example request:

DELETE

CURL :

```
curl
'http://localhost:8160/v1/threads/{threadToken}?workspaceId=-1' \
--request DELETE \
--header 'Api-Token: YOUR_SECRET_TOKEN'
```

C#:

```
using System.Net.Http.Headers;
```


Main
More
Proxy

☒ Redirect

☐ Use original url

Headers:
User settings

Request headers:

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>

Deleting multiple threads

Description: This method deletes multiple execution threads.

Request parameters:

Parameter	Type	Format	Default	Description
workspaceId	integer	int64	-1	Workspace identifier. -1 means the default workspace.
body	JSON-Array			Target execution thread tokens Example: ["uuid1", "uuid2"]

Note:

The list of threads to delete should be specified as the array in the request body using the following format:

```
[  
  "threadToken1",  
  "threadToken2"  
]
```

Example request:**DELETE****CURL :**

```
curl  
'http://localhost:8160/v1/threads/delete_bulk?workspaceId=-1' \  
  --request DELETE \  
  --header 'Content-Type: application/json' \  
  --header 'Api-Token: YOUR_SECRET_TOKEN' \  
  --data '[  
    "threadToken1",  
    "threadToken2"  
  ]'
```

C#:

```
using System.Net.Http.Headers;  
var client = new HttpClient();  
var request = new HttpRequestMessage  
{  
    Method = HttpMethod.Delete,  
    RequestUri = new Uri(  
"http://localhost:8160/v1/threads/delete_bulk?workspaceId=-1"),  
    Headers =  
    {  
        { "Api-Token", "YOUR_SECRET_TOKEN" },  
    },  
    Content = new StringContent("[\n  " +  
"\n\"threadToken1\", " +  
"\n\"threadToken2\", " +  
"\n]")  
    {  
        Headers =  
        {  

```

```

        ContentType = new
MediaTypeHeaderValue("application/json")
    }
}
};
using (var response = await client.SendAsync(request))
{
    response.EnsureSuccessStatusCode();
    var body = await response.Content.ReadAsStringAsync();
    Console.WriteLine(body);
}

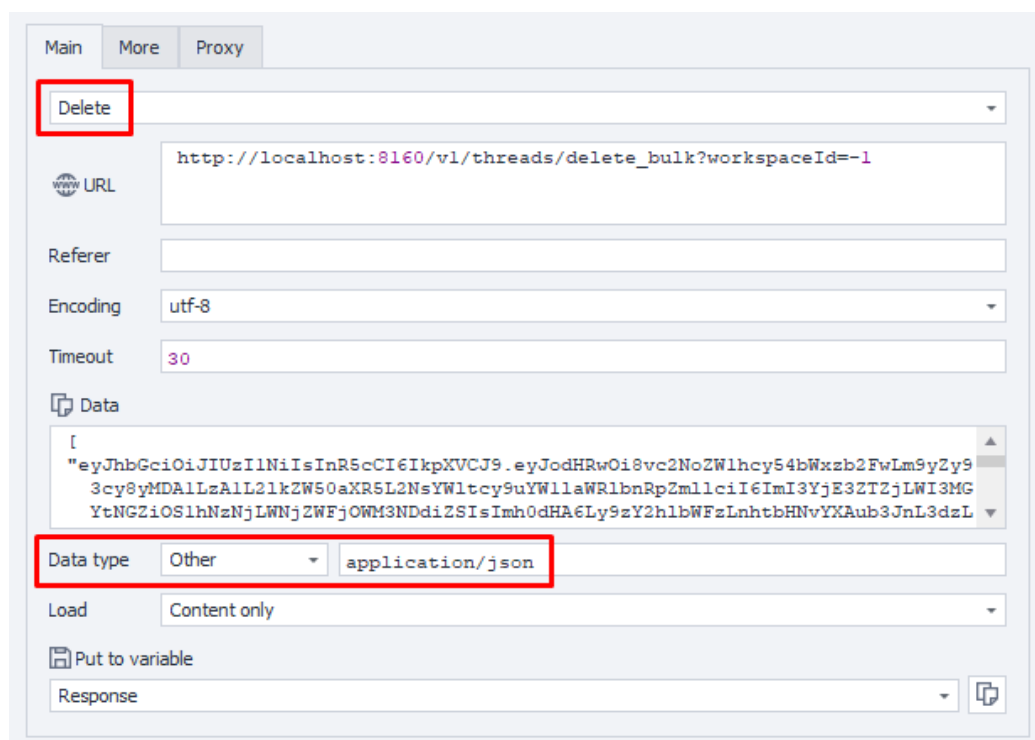
```

Curl:

`http://localhost:8160/v1/threads/delete_bulk?workspaceId=-1`

Body:

`["threadToken1", "threadToken2"]`



Additionally:

User-Agent: {-Profile.UserAgent-}

Api-Token: Token from **UserArea2**.

MainMoreProxy

☒ Redirect

☐ Use original url

Headers: User settings

Request headers: ?

User-Agent: {-Profile.UserAgent-}
Api-Token: {-Variable.Token-}

☒ Use Cookie Container

Response API:

Response code	Result	Response
200 OK	Success	
500 Error	Internal Server Error	<pre>{ "message": null }</pre>