API Doc Publication

Table of Contents

1. Profile API

This section provides a high-level overview of our API, covering its functionality and applications. Currently, we have enabled Profile API using which, you can call traits or profile attributes, consent or marketing preferences of an identity's user profile. This API can be used to provide personalised experiences for your users across any channel.

> The Profile API is accessible to all Zeotap accounts with no restrictions on the number of requests per second (RPS).
> However, for the limitations on the payload size, refer here.

Key Features of Profile API

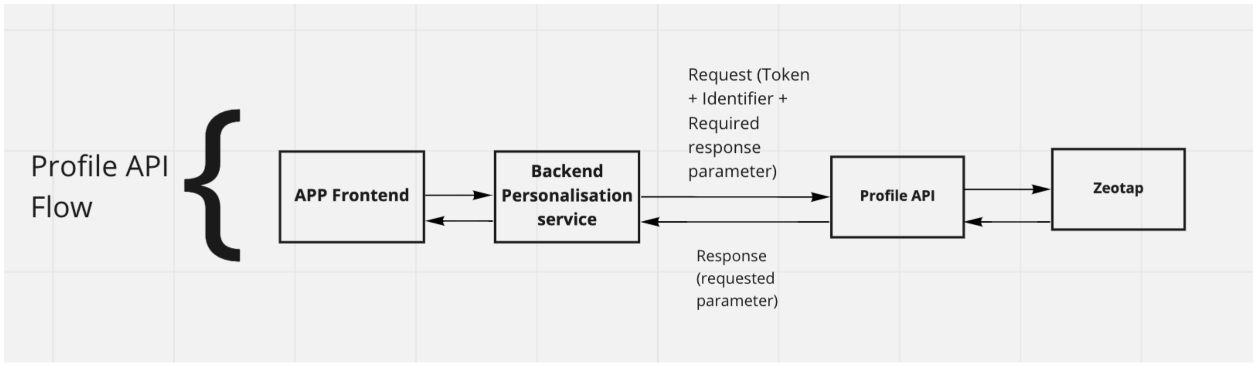
The following are some of the key features of the Profile API:

- **Fast response times** - Fetch traits from a user profile under 200ms
- **Real-time data** - Query streaming data on the user profile
- **One identity** - Query an end user's interactions across web, mobile, server, and third-party touch-points
- **Rich data** - Query `IDs`, `Profile`, `Consent`, `Marketing Preferences` and `Calculated Attributes`
- **Any external ID** - The API supports query from `user_id`, `advertising IDs`, `anonymous_id` and `custom external IDs`

Profile API Workflow

As a best practice, avoid making direct queries to the Profile API from your app's front end. Instead, create a dedicated back-end service to successfully implement personalisation using the Profile API. This service can be integrated into your existing backend framework or operate as a standalone service. It serves as a mediator between your front end and the Profile API, thereby enhancing efficiency and security. Refer to the flow diagram below to understand the Profile API Workflow better.

API Doc Publication


Profile API Flow
APP Frontend
Backend Personalisation service
Request (Token + Identifier + Required response parameter)
Profile API
Zeotap
Response (requested parameter)

The following steps outline how the Profile API works:

1. Your app client requests specific personalisation data, such as product recommendations, from your personalisation service, providing the user's identity.
2. The back-end personalisation service interacts with the Profile API, determining the appropriate app behaviour based on the user's profile information.
3. Subsequently, the personalisation service exclusively furnishes your app's front end with the essential information required to implement the desired personalisation.

Get Started

To start using Profile API, reach out to your Customer Success Manager or Zeotap Support Team to activate the feature and get the necessary Authentication Token, which in this case is the API Key associated with your Organisation. Ensure that you familiarise yourself with critical technical concepts, such as creating an authorisation header, payload body, which is crucial for executing an API call. In addition, ensure that your understand the status codes and responses that indicate the success or failure of an API request. Once you have understood these fundamentals, you can proceed to make your first API call using the Base URL along with the operation that you want to perform. Below section helps you to understand the various operations available in the system.

Manage User Profiles

You can use our Profile API to perform the following operation:

→ Fetch User Profiles

API Doc Publication

Use this POST request to read or look up a user profile in the Zeotap system.

→ Create or Update User Profiles

Use this POST request to create or update a user profile in the Zeotap system.

→ Delete User Profiles

Use this POST request to delete a user profile from the Zeotap system.

Base URL

Use the appropriate production endpoint based on your preferred data transport method.

- For HTTPS Requests, use the endpoint – https://api.zeotap.com
- For mTLS Requests, use the endpoint – https://mtls-api.zeotap.com

Authentication

Currently, Zeotap supports API Key authentication only. However, we extend support to additional authentication schemes based on the need.

API Key Authentication - This method utilises a unique identifier (API Key) assigned to an Organisation to authenticate API requests. This API Key is generated by Zeotap when your Organisation is created within the Zeotap system. You can include this key in the HTTP header or URL parameters of API requests, which enables the Zeotap server to validate and authorise the request.

Data Transport Protocols/Methods

The following are the protocols/methods supported for secure data transmission:

- **HTTPS (Hypertext Transfer Protocol Secure)**: Encrypts data sent between your application and the Zeotap server, ensuring secure communication over the internet.

- **mTLS (Mutual Transport Layer Security)**: Allows both your application and the Zeotap server to authenticate each other and establish an encrypted connection, thereby enhancing security for data transmission. For information about how to configure mTLS method for data transport, refer here.

Quotas and Rate Limits

Currently, there are no limitations on the number of API calls to the Zeotap server. Moreover, the platform has the capability to autoscale to handle fluctuations or spikes in API requests.

Payload Size

The following are the limitations on the number of immutable IDs (unique profiles) that can be managed through Profile API:

OPERATION	DESCRIPTION
Read	Allows to fetch up to 5 immutable IDs per API call
Write	Allows to create/update one immutable ID per API call
Delete	Allows to delete up to 400 IDs per API call

Status Codes and Responses

We use the conventional HTTP response codes to indicate the success or failure of an API request. The following table lists the status/error codes that are returned by the API requests:

STATUS/ERROR CODES	DESCRIPTION
`200 - OK`	This is the response when your request is accepted and processed successfully.
`204 - No Content`	This is the response when your request is accepted.
`400 - Bad Request`	This is the response when your request is unacceptable, which is often due to a required parameter that is missing.
`401 - Unauthorized`	This is the response when your request is not processed due to inadequate user permissions or invalid access token.
`403 - Forbidden`	This is the response when you are forbidden from accessing a valid URL.
`404 - Not Found`	This is the response when you request a resource that does not exist.
`429 - Too Many Requests`	This is the response when too many requests hit the API too quickly.
`500, 502, 503, 504 - Server Errors`	This is the response when there is an issue at Zeotap's end.

`400 - Bad Request`
 	
`Response: Search failed as no Identifier found in the request`
	This is the response when you send a null or empty Id value.

STATUS/ERROR CODES	DESCRIPTION

`400 - Bad Request`
 	
`Response : Request failed as no Org ID was found in request`
	This is the response when you send a request without the Org ID key or value.

Best Practices and Recommendations

The following are the best practices and recommendations that we suggest:

- We recommend you invoke the Profile API once per session, only. As the response is unlikely to change, even if the API is called multiple times, you can cache the response locally for subsequent use within the session. However, note that calling the API multiple times is still counted against your usage metrics.

- To avoid unnecessary count against your usage metrics, we advise you to add a check to prevent calling the Profile API when there is no ID to look up.

- We recommend you use a separate API token for each interface. This ensures that each interface is uniquely identified and managed separately. Using separate API tokens also enhances the security of your system by limiting access to individual interfaces when one token is compromised.

- For user lookups, we recommend you use one of the immutable IDs or primary IDs that you have selected for ID resolution. This ensures that the user is identified uniquely and the lookup is consistent across the different systems.

- To ensure security, we do not recommend you implement Profile API on a web interface. Instead, we recommend you implement the API remotely using a service layer. This approach minimises the risk of unauthorised access and other security vulnerabilities that may arise when implementing the API on a web interface.

- To call a specific attribute of a user, we recommend you use the `Fetch` node. This allows you to retrieve only the required attribute instead of calling all user attributes on the client side.

Related Topics

- For information about **how to use Read API** to read or look up a user profile in Zeotap, refer here.
- For information about **how to use Write API** to create or update a user profile in Zeotap, refer here.
- For information about **how to use Delete API** to delete a user profile in Zeotap, refer here.
- For **Read API - Sample Requests and Responses**, refer here.
- For **Write API - Sample Requests and Responses**, refer here.
- For **Delete API - Sample Requests and Responses**, refer here.

Next Topic → Read API

← Previous Page
Understand Profile API through Use Cases
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket

👍👎1.1. Understand Profile API through Use Cases

Main Topic ↑ Profile API

The following are some of the use cases of Profile API:

Use Case 1: Site Personalisation

The Profile API can be integrated into your website's backend to fetch user profile attributes and preferences. When a user visits your site, the API retrieves their profile data, including demographics, interests, and consent choices. Using this data, your site can dynamically tailor content and recommendations, ensuring a personalised browsing experience. For example, it can display targeted advertisements, suggest relevant articles or products and customise the user interface based on individual preferences.

Use Case 2: Product Recommendations

The Profile API can be instrumental in providing personalised product suggestions to users. By accessing user profile attributes and preferences, the API can identify products or services that align with their interests and needs. Subsequently, your application or e-commerce platform can utilise this information to generate real-time product recommendations. For example, if a user has shown a preference for electronics in their profile, the API can recommend specific gadgets or accessories when they browse your online store. This enhances user engagement and can lead to increased sales and customer satisfaction.

Use Case 3: Retrieving Profile or Consent Data for Agent Calls

The Profile API facilitates the retrieval of user profile and consent information for agent-assisted interactions. When a customer contacts your customer support or sales team, the API can provide a comprehensive view of the customer's profile and their consent status. Your agents can access this information during the call, ensuring they have a complete understanding of the customer's preferences and history. This enables them to provide more personalised assistance, make relevant product recommendations and address customer concerns effectively.

Next Topic → Read API

Profile API
← Previous Page
Fetch User Profiles
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket

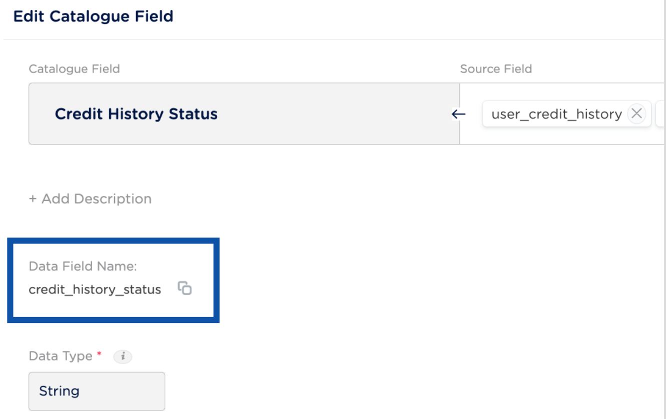
1.2. Fetch User Profiles

Main Topic ↑ Profile API

You can read or look up a user profile in the Zeotap system by making a valid `POST` call with the details provided in the table below.

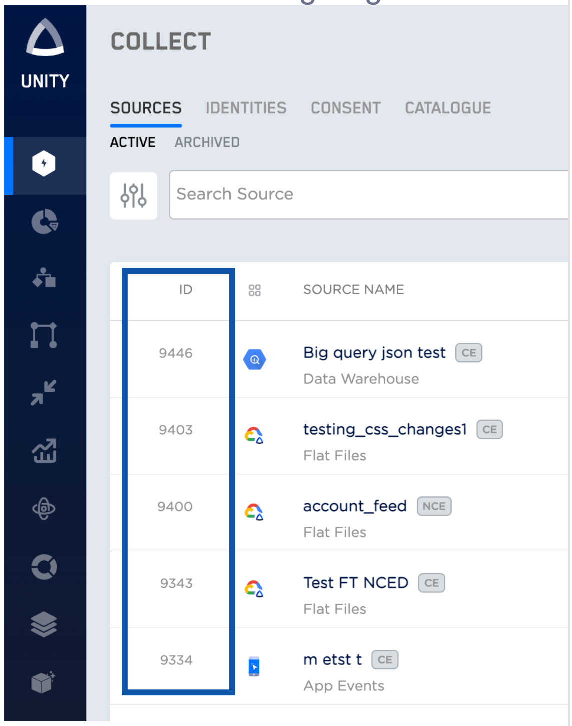
API Doc Publication

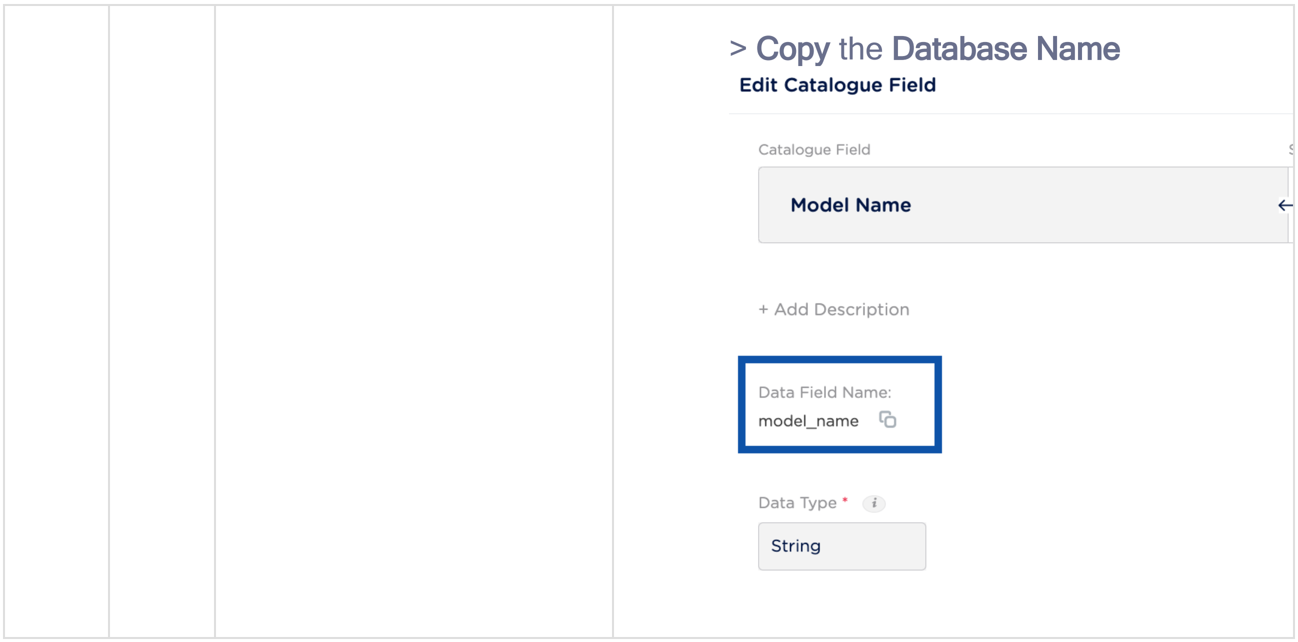
HTTP Request Method	POST		
Endpoint URL	**For HTTPS** - `https://api.zeotap.com/cdp/v1/users/_search`		
	For mTLS - `https://mtls-api.zeotap.com/cdp/v1/users/_search`		
Headers	Content-Type	application/json	
	apikey	<api_key_associated_with_your_orgId >	
Body			
Parameter	**Mandatory**	**Type**	**Description**
`orgId`	**Yes**	integer(int64)	This is the ID assigned to your account while onboarding.
`search`	**Yes**	Map<String, List> Allowed keys - Any ID Type attribute ℹ We recommend th the immutable IDs for ID Resolution in yo organisation.	The ID based on which the search is performed.
`regions`	**Optional**	Array[String]	Specify the Region in which the data is to be searched.

			If not specified, then the Profile look-up is done across all the regions available for your account.
fetch	**Optional** If fetch is not present, then all the details are returned.	Map<String, List> Allowed keys : [ids, profile, consent, mkt_preferences, calc_attributes]	The name of the keys for which the values are to be fetched. The attribute name must be the same as the database name in your Zeotap organisation. You can find the database name as follows: **Collect** > **Catalogue** > **Edit** the required attribute > **Copy** the database name  **Edit Catalogue Field** Catalogue Field: Credit History Status Source Field: user_credit_history + Add Description Data Field Name: credit_history_status Data Type *: String

fetch Nce	**Optional**	Map<String, List>
		Allowed keys :
		"fetch_source": {
		    "dsId1":["join_key1"],
		    "dsId2":["join_key1","join_key2"]
		  }
		Where,
			Specify the Data Source IDs and Join Keys for which the values are to be fetched. Based on the Data Source ID and Join Key combination included in the `fetch_source` node, we return the corresponding NCE data available in the system.
			To learn how to obtain the Data Source ID and Join Key, refer below.

API Doc Publication

		• dsId represents the Data Source IDs. • join_key represents the Database name of the customer attribute, with which the NCE source is joined.	• You can obtain the Data Source ID from the source listing Page.  UNITY COLLECT SOURCES IDENTITIES CONSENT CATALOGUE ACTIVE ARCHIVED Search Source ID — SOURCE NAME 9446 — Big query json test CE — Data Warehouse 9403 — testing_css_changes1 CE — Flat Files 9400 — account_feed NCE — Flat Files 9343 — Test FT NCED CE — Flat Files 9334 — m etst t CE — App Events • You can find the Join Key by clicking the particular NCE source and going to the **Catalogue Mapping** tab. In addition, you can find the database name of this attribute as follows: **Collect** > **Catalogue** > **Edit** the required attribute



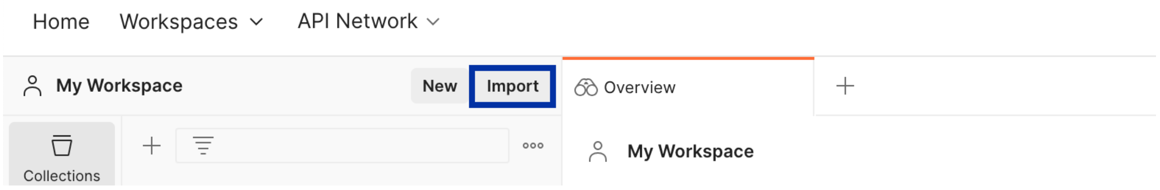
Import Curl for Read API

You can import the curl given below into the Postman tool and fetch endpoint URL, headers and payload body from the curl automatically. To do so, perform the following steps:

•

1. Open the Postman tool on your system.

2. Click **Import** as shown below.



3. Copy the sample Read API curl given below after you replace the `API Key` value and the payload body in the curl.

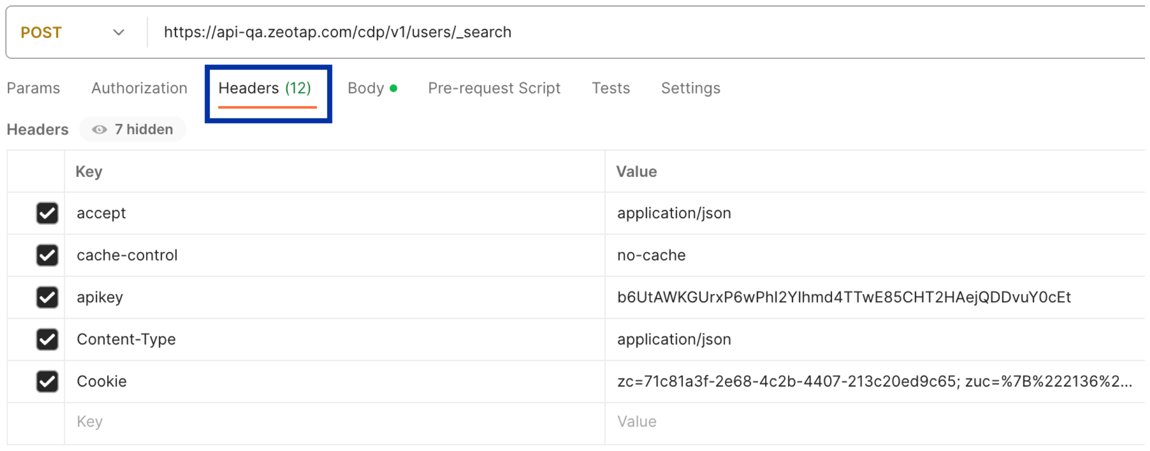
Code

```
curl --location 'https://api.zeotap.com/cdp/v1/users/_search' \
--header 'accept: application/json' \
--header 'cache-control: no-cache' \
```

API Doc Publication

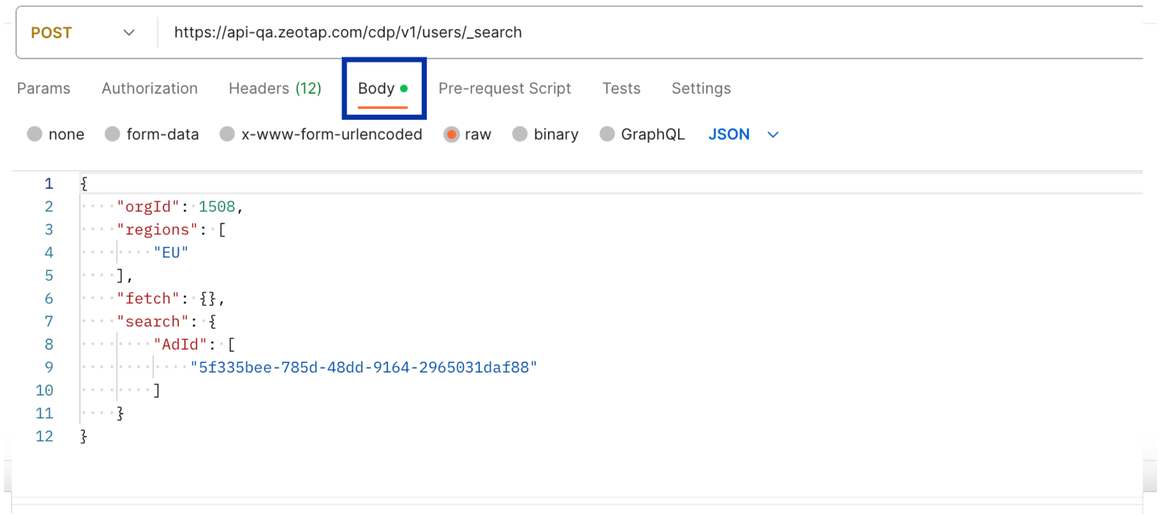
```
--header 'apikey: api_key associated_with_the_org_id' \
--header 'Content-Type: application/json' \
--header 'Cookie: zc=71c81a3f-2e68-4c2b-4407-213c20ed9c65;
zuc=%7B%222136%22%3A%7B%22v%22%3A%22ALL%22%2C%22p%22%3A%22ALL%22%2C%22t%22%3
\
--data '{
    "orgId": 1508,
    "regions": [
        "EU"
    ],
    "fetch": {},
    "search": {
        "AdId": [
            "5f335bee-785d-48dd-9164-2965031daf88"
        ]
    }
}'
```

4. In the window that appears on Postman, paste the copied curl into the provided field. This automatically fetches the endpoint URL, headers and body from the curl.



```
POST https://api-qa.zeotap.com/cdp/v1/users/_search
```

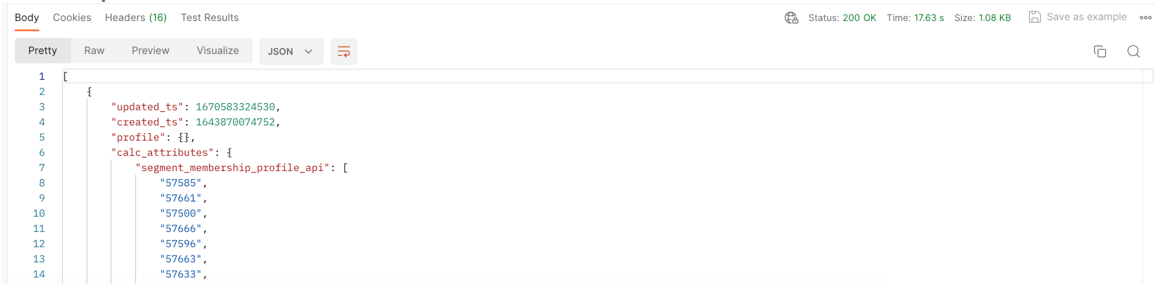
	Key	Value
✓	accept	application/json
✓	cache-control	no-cache
✓	apikey	b6UtAWKGUrxP6wPhl2YIhmd4TTwE85CHT2HAejQDDvuY0cEt
✓	Content-Type	application/json
✓	Cookie	zc=71c81a3f-2e68-4c2b-4407-213c20ed9c65; zuc=%7B%222136%2...
	Key	Value



```
POST https://api-qa.zeotap.com/cdp/v1/users/_search
```

```
{
    "orgId": 1508,
    "regions": [
        "EU"
    ],
    "fetch": {},
    "search": {
        "AdId": [
            "5f335bee-785d-48dd-9164-2965031daf88"
        ]
    }
}
```

5. Click **Send**. If your POST call is valid, then you receive **Status: 200 OK**, along with the response as shown below.



```
Status: 200 OK  Time: 17.63 s  Size: 1.08 KB
```

```
[
    {
        "updated_ts": 1670583324530,
        "created_ts": 1643870074752,
        "profile": {},
        "calc_attributes": {
            "segment_membership_profile_api": [
                "57585",
                "57661",
                "57500",
                "57666",
                "57596",
                "57663",
                "57633",
```

Sample Request and Response Payloads

This section presents some sample request and response payloads for Read API.

Sample Request 1 (To fetch all the attributes)

Code

```
{
  "orgId": 1234,
  "regions": ["EU"],
  "search": {
    "email_sha256_lowercase": [
      "9dff3928e6f62ef808"
    ]
  }
}
```

Sample Response 1

Code

```
[
  {
    "updated_ts": 1635945855983,
    "created_ts": 1635945855983,
    "profile": {
      "age": {
        "_ts": "1623176905769",
        "age": "21"
      },
      "user_city": {
        "_ts": "1623176905769",
        "user_city": "agra"
      }
    },
    "ids": {
      "AdId": [
        {
          "AdId": "pbmaid2",
          "_ts": "1623160405332"
        }
      ],
      "email": [
        {
          "_ts": "1623176905769",
          "email_sha256_lowercase": "9dff3928e6f62ef808",
          "email_sha256_uppercase": "130ade38c2ebd580ae"
        }
      ]
    },
    "mkt_preferences": {},
    "consent": {},
    "region": "EU",
    "ucid": "dee32bf9-ceeb-4cae-885a-376a5f052cfc"
  }
]
```

When no key is specified in the request, the API returns all the attributes. To retrieve a specific attribute, use the `Fetch` node.
However, NCE is a fetch-only attribute that returns information based on the `fetchNce` request. Therefore, use the specific Data Source ID and Join Key combination for a particular NCE data to be returned. For more information about how to fetch NCE data, refer to Sample Request 4 and Sample Response 4.

Sample Request 2 (To fetch specific attributes using the fetch node)

Code

```
{
  "orgId": 1234,
  "regions": ["EU"],
  "fetch": {
    "ids": [
      "AdId"
    ],
    "profile": [
      "age"
    ],
    "consent": [
      "gdpr_advertising",
      "gdpr_marketing"
    ],
    "mkt_preferences": [
      "telemarketing"
    ]
  },
  "search": {
    "AdId": [
      "pbmaid1"
    ]
  }
}
```

Sample Response 2

Code

```
[
  {
    "updated_ts": 1635945855983,
    "created_ts": 1635945855983,
    "profile": {
      "age": {
        "_ts": "1623176905769",
        "age": "27"
      }
    },
    "ids": {
      "AdId": [
        {
          "AdId": "pbmaid1",
          "_ts": "1623160405332"
        }
      ]
    },
    "mkt_preferences": {
      "telemarketing": {
        "datasourceId": "5",
        "datasetId": "6530",
        "type": "telemarketing",
        "value": "market",
        "_ts": "1635945855983"
      }
    },
    "consent": {
      "gdpr_advertising": {
        "datasourceId": "5",
        "datasetId": "6530",
        "type": "gdpr_advertising",
        "value": "yes",
        "ttl": "63072000",
        "_ts": "1635945855983"
      },
      "gdpr_marketing": {
        "datasourceId": "5",
        "datasetId": "6530",
        "type": "gdpr_marketing",
        "value": "yes",
        "ttl": "63072000",
        "_ts": "1635945855983"
      }
    },
    "region": "EU",
    "ucid": "8019c3f7-844f-4388-ab52-7f73f3db7c4f"
  }
]
```

Ensure that you take care of the following points, while using the fetch node:

- The keys that are absent for the searched user, cannot be fetched. For example, in the above sample, gender in profile was not found. Therefore, those are not part of the response.
- The Profile API can return only ID, profile, consent, marketing preference, calculated attributes and NCE attributes as part of its response.

Sample Request 3 (To fetch system calculated attribute like segment membership)

Code

```
{
    "orgId": 1234,
    "regions": ["EU"],
    "search": {
        "ZCookie": [
            "10cccd2d-9e97-4f19-7ae7-5e14abf49bdd"
        ]
    },
    "fetch":{
        "profile":["Age_Group"],
        "ids":["zcookie_amazon","Tapad"],
        "calc_attributes": ["segment_membership"],
        "mkt_preferences":[]
    }
}
```

Sample Response 3

Code

```
[
    {
          "updated_ts": 1669870508762,
          "created_ts": 1669870274612,
          "profile": {
              "Age_Group": {
                  "Age_Group": "31-31",
                  "_ts": "1669816844837"
              }
          },
          "calc_attributes": {
              "segment_membership": [
                  "62993",
                  "62992"
              ]
          },
          "ids": {
              "zcookie_amazon": [
                  {
                      "zcookie_amazon": "d057ecf8-7c76-462c-6032-
c58a9dfe6526",
                      "_ts": "1669816844717"
                  }
              ],
              "Tapad": [
                  {
                      "Tapad": "6921ea20-f268-40e5-98ca-c4b259ce949b",
                      "_ts": "1669816844774"
                  }
              ]
          },
          "mkt_preferences": {},
          "region": "EU",
          "ucid": "385d56f4-1bc4-4a96-8636-986850378616"
      }
]
```

Sample Request 4 (To fetch NCE Data)

You can use this request to fetch NCE data associated with particular data sources using the Join Keys. Based on the Data Source ID and Join Key combination included in the `fetch_source` node, we return the corresponding NCE data available in the system. In the below example, `10733` is the Data Source ID and `car_name` and `prodname` are the Join Keys.

Code

```
{
    "orgId": 2226,
    "regions": ["EU"],
    "fetch": {
        "ids": [],
        "profile": [],
        "consent": [],
        "mkt_preferences": [],
        "calc_attributes": []
    },
    "fetchNce": {
        "fetch_source":{
            "10733":["car_name","prodname"]
        }
    },
    "search": {
        "AdId": [
            "nce_user_1"
        ]
    }
}
```

Sample Response 4

Based on the above request, the NCE data associated with Data Source ID `10733` and the Join Keys `car_name` and `prodname` is displayed within the `nce_attributes` node as shown in the sample response below. Note that the NCE data is fetched for the following two combinations:

- Data Source ID `10733` with the Join Key `car_name`
- Data Source ID `10733` with the Join Key `prodname`

Code

```
[
  {
    "updated_ts": 1691649060229,
    "created_ts": 1691647767022,
    "profile": {
      "car_name": {
        "_ts": "1691649060212",
        "car_name": [
          "BMW X1",
          "AMG G63"
        ]
      },
      "prodname": {
        "_ts": "1691649060212",
        "prodname": "Porche 911"
        }
      },
      "calc_attributes": {},
      "ids": {
        "AdId": [
          {
            "AdId": "nce_user_1",
            "_ts": "1691649060201"
          }
        ]
      },
      "mkt_preferences": {},
      "consent": {},
      "region": "EU",
      "nce_attributes": {
        "car_name": {
          "10733": [
            {
              "_zeotap_ts": "1691643997785000",
              "car_model": "BMW X1",
              "datasetId": "10733",
              "feed_prod_cat": "premium car",
              "feed_prod_color": "blue",
              "feed_prod_price": "2lac",
            },
            {
              "_zeotap_ts": "1691644053145000",
              "car_model": "AMG G63",
              "datasetId": "10733",
              "feed_prod_cat": "premium suv",
              "feed_prod_color": "black",
              "feed_prod_price": "45lac",
            }
          ]
        },
        "prodname": {
          "10733": [
            {
              "_zeotap_ts": "1691644053287000",
              "car_model": "Porche 911",
              "datasetId": "10733",
              "feed_prod_cat": "premium sports car",
              "feed_prod_color": "green",
              "feed_prod_price": "89lac",
            }
          ]
        }
      },
      "ucid": "4a270d75-ad53-440b-bf17-331dda4d7191"
    }
  ]
```

Status Codes and Responses

The table below lists the different scenarios and their corresponding status codes and responses.

STATUS/ERROR CODES	DESCRIPTION
200 - OK	When the request is accepted and processed successfully.
204 - No Content	When invalid data is passed. For example, email_sha256_lowercases is passed instead of email_sha256_lowercase.
400 - Bad Request	When the request is unacceptable, often due to missing a required parameter.
401 - Unauthorized	When invalid access key is passed.

Best Practices

For Read API Best Practices and recommendations, refer here.

Next Topic → Write API

Understand Profile API through Use Cases
← Previous Page
Create or Update User Profiles
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket

1.3. Create or Update User Profiles

Main Topic ↑ Profile API

You can create or update a user profile in the Zeotap system by making a valid `POST` call with the details provided in the table below.

HTTP Request Method	`POST`		
Endpoint URL	**For HTTPS** - `https://api.zeotap.com/cdp/v1/users`		
	For mTLS - `https://mtls-api.zeotap.com/cdp/v1/users`		
Headers	Content-Type	application/json	
	apikey	<api_key_associated_with_your_orgId>	
Body			
Parameter	**Mandatory**	**Type**	**Description**
`orgId`	Yes	integer(int64)	This is the ID assigned to your account while onboarding.
`region`	Yes	String	Specify the Region in which the data is to

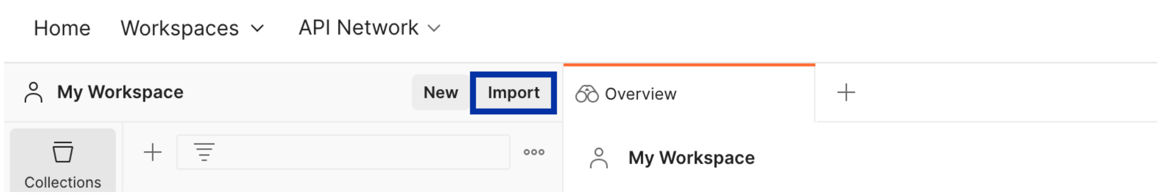
			be searched.
ids	**Yes**	Allowed keys - Any ID Type attribute ℹ We recommend that you use one of the immutable IDs/Primary IDs selected for ID Resolution in your Zeotap organisation.	The IDs for which the values are to be updated.
upsert	**Yes**	Allowed keys - [profile, consent, mktPreferences]	The name of the keys for which the values are to be updated.
ucid	Optional • If ucid is present, then the user preference is updated. • If ucid is not present, then a new user is created.	String	The Unique Customer ID assigned to the user by Zeotap.

Import Curl for Write API

You can import the curl given below into the Postman tool and fetch endpoint URL, headers and payload body from the curl automatically. To do so, perform the following steps:

-
  1. Open the Postman tool on your system.

  2. Click **Import** as shown below.

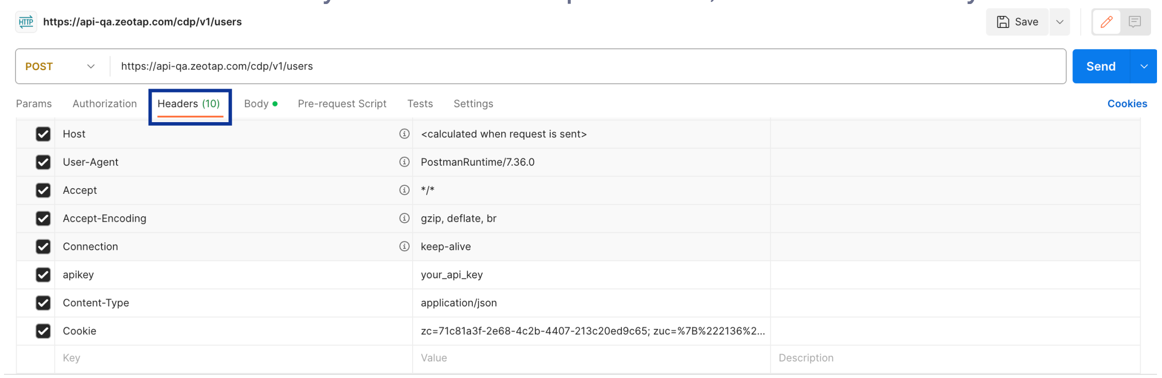


  3. Copy the sample Write API curl given below after you replace the `API Key` value and the payload body in the curl.

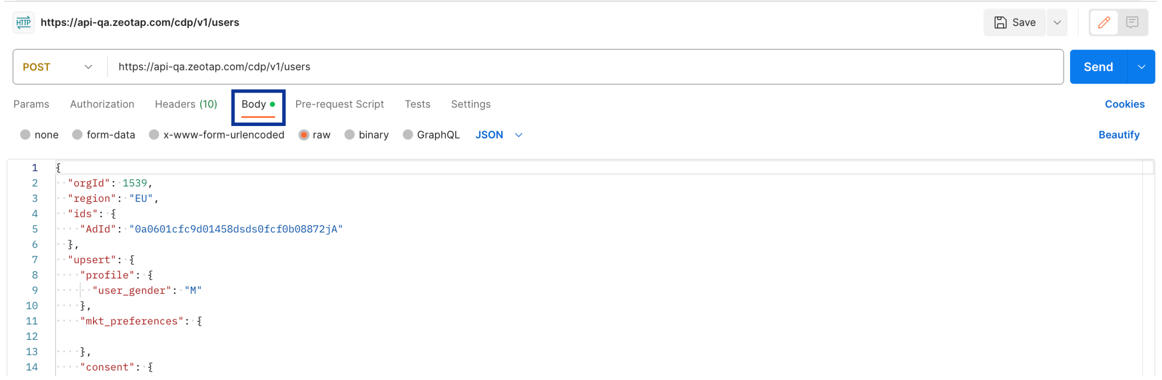
Code

```
curl --location 'https://api-qa.zeotap.com/cdp/v1/users' \
--header 'apikey: your_api_key' \
--header 'Content-Type: application/json' \
--header 'Cookie: zc=71c81a3f-2e68-4c2b-4407-213c20ed9c65;
zuc=%7B%222136%22%3A%7B%22v%22%3A%22ALL%22%2C%22p%22%3A%22ALL%22%2C%22t%22%3
\
--data '{
  "orgId": 1539,
  "region": "EU",
  "ids": {
    "AdId": "0a0601cfc9d01458dsds0fcf0b08872jA"
  },
  "upsert": {
    "profile": {
      "user_gender": "M"
    },
    "mkt_preferences": {

    },
    "consent": {

    }
  }
}'
```

4. In the window that appears on Postman, paste the copied curl into the provided field. This automatically fetches the endpoint URL, headers and body from the curl.



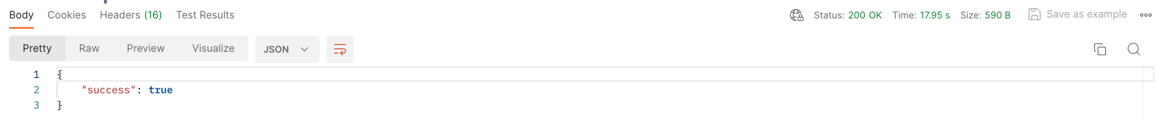
	Key	Value	Description
✓	Host	<calculated when request is sent>	
✓	User-Agent	PostmanRuntime/7.36.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	apikey	your_api_key	
✓	Content-Type	application/json	
✓	Cookie	zc=71c81a3f-2e68-4c2b-4407-213c20ed9c65; zuc=%7B%222136%2...	



POST https://api-qa.zeotap.com/cdp/v1/users

```
{
  "orgId": 1539,
  "region": "EU",
  "ids": {
    "AdId": "0a0601cfc9d01458dsds0fcf0b08872jA"
  },
  "upsert": {
    "profile": {
      "user_gender": "M"
    },
    "mkt_preferences": {

    },
    "consent": {
```

5. Click **Send**. If your POST call is valid, then you receive **Status: 200 OK**, along with the response as shown below.



Status: 200 OK Time: 17.95 s Size: 590 B

```
{
    "success": true
}
```

Sample Request and Response Payloads

This section presents some sample request and response payloads for Write API.

Sample Request 1 (Update User)

As the UCID key is provided, the user is updated.

- apikey : <Token>
- The field names may differ based on your organisation’s catalogue.

Code

```
{
  "orgId": 1234,
  "region": "EU",
  "ucid": "d7a6bc75-2c87-4650-91dc-c212771866c9",
  "ids": {
    "email_sha256_lowercase": "0a0601cfc9d014580fcf0b0"
  },
  "upsert": {
    "profile": {
      "user_gender": "M"
    },
    "mkt_preferences": {
      "telemarketing": "market",
      "sms_optin": "yes"
    },
    "consent": {
      "gdpr_advertising": "yes"
    }
  }
}
```

Sample Response 1

Code

```
{
    "success": true
}
```

Sample Request 2 (Create User)

As the UCID key is not present, a new user is created.

- apikey : <Token>
- The field names may differ based on your organisation’s catalogue.

Code

```
{
  "orgId": 1234,
  "region": "EU",
  "ids": {
    "email_sha256_lowercase": "0a0601cfc9d014580fcf0b0"
  },
  "upsert": {
    "profile": {
      "user_gender": "M"
    },
    "mkt_preferences": {
      "telemarketing": "market",
      "sms_optin": "yes"
    },
    "consent": {
      "gdpr_advertising": "yes"
    }
  }
}
```

Sample Response 2

Code

```
{
    "success": true
}
```

Best Practices

For Write API Best Practices and recommendations, refer here.

Next Topic ⇥ Delete API

Fetch User Profiles
← Previous Page
Delete User Profiles
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket

👍👎1.4. Delete User Profiles

Main Topic ⇡ Profile API

You can delete a user profile from the Zeotap system by making a valid `POST` call with the details provided in the table below.

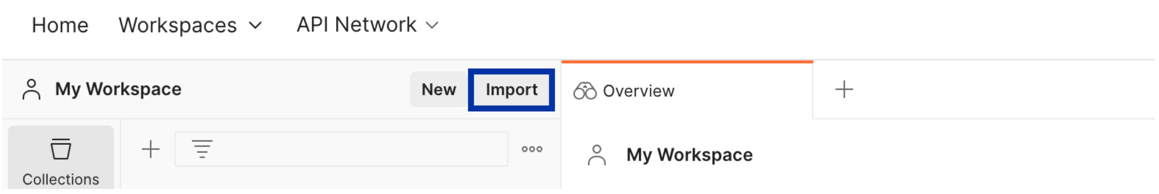
HTTP Request Method	POST		
Endpoint URL	**For HTTPS** - `https://api.zeotap.com/cdp/v1/users/_delete`		
	For mTLS - `https://mtls-api.zeotap.com/cdp/v1/users/_delete`		
Headers	Content-Type	application/json	
	apikey	<api_key_associated_with_your_orgId>	
Body			
Parameter	**Mandatory**	**Type**	**Description**
orgId	Yes	integer(int64)	This is the ID assigned to your account while onboarding.
region	Yes	String	Specify the Region in which the data is to be searched

			and deleted.
`ids`	Yes	Map<String, List> Allowed keys - Any ID Type attribute ℹ • We recommend that you use one of the immutable IDs/Pr IDs selected for ID Resolutic your Zeotap organisation. • In a delete request, you can only one ID attribute name t perform look-up and delete. • A maximum of 400 Ids can b per request.	The ID based on which the search is performed.
`sendEmailNotification`	No	Boolean	This field is set to 'false' by default. To notify the Admin for every deleted profile, set this to 'true'.

Import Curl for Delete API

You can import the curl given below into the Postman tool and fetch endpoint URL, headers and payload body from the curl automatically. To do so, perform the following steps:

-
  1. Open the Postman tool on your system.
  2. Click **Import** as shown below.

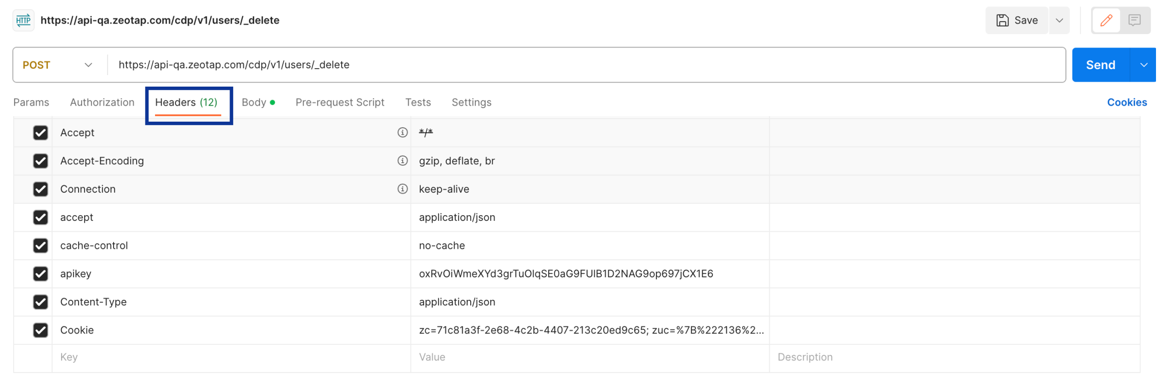
     

  3. Copy the sample Write API curl given below after you replace the `API Key` value and the payload body in the curl.

Code

```
curl --location 'https://api-qa.zeotap.com/cdp/v1/users/_delete' \
--header 'accept: application/json' \
--header 'cache-control: no-cache' \
--header 'apikey: oxRvOiWmeXYd3grTuOlqSE0aG9FUlB1D2NAG9op697jCX1E6' \
--header 'Content-Type: application/json' \
--header 'Cookie: zc=71c81a3f-2e68-4c2b-4407-213c20ed9c65;
zuc=%7B%222136%22%3A%7B%22v%22%3A%22ALL%22%2C%22p%22%3A%22ALL%22%2C%22t%22%3
\
--data '{
    "orgId": 1539,
    "region": "EU",
    "ids": {
        "AdId": [
            "ddd682ec-bbe9-4705-bb02-3b9eacaf93bf"
        ]
    },
    "sendEmailNotification": "false"
}'
```

4. In the window that appears on Postman, paste the copied curl into the provided field. This automatically fetches the endpoint URL, headers and body from the curl.

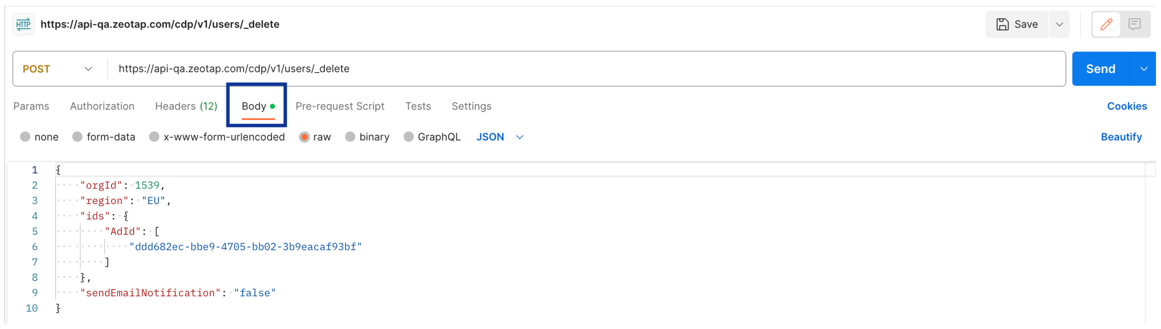


https://api-qa.zeotap.com/cdp/v1/users/_delete

POST https://api-qa.zeotap.com/cdp/v1/users/_delete

Params · Authorization · Headers (12) · Body · Pre-request Script · Tests · Settings · Cookies

	Key	Value	Description
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	accept	application/json	
✓	cache-control	no-cache	
✓	apikey	oxRvOiWmeXYd3grTuOlqSE0aG9FUIB1D2NAG9op697jCX1E6	
✓	Content-Type	application/json	
✓	Cookie	zc=71c81a3f-2e68-4c2b-4407-213c20ed9c65; zuc=%7B%222136%2...	



https://api-qa.zeotap.com/cdp/v1/users/_delete

POST https://api-qa.zeotap.com/cdp/v1/users/_delete

Params · Authorization · Headers (12) · Body · Pre-request Script · Tests · Settings · Cookies

none · form-data · x-www-form-urlencoded · [✓ raw] · binary · GraphQL · JSON · Beautify

```
{
    "orgId": 1539,
    "region": "EU",
    "ids": {
        "AdId": [
            "ddd682ec-bbe9-4705-bb02-3b9eacaf93bf"
        ]
    },
    "sendEmailNotification": "false"
}
```

5. Click **Send**. If your POST call is valid, then you receive **Status: 200 OK**, along with the details of the deleted profile.

Sample Request and Response Payloads

This section presents some sample request and response payloads for Delete API.

Points to Note:

- Auth token: Bearer `RBAC_TOKEN`
- The field names may differ based on your organisation's catalogue.

Sample Request 1 - (Using Primary ID)

Code

```
{
    "orgId": 1539,
    "region": "EU",
    "ids": {
        "Google": ["92ed24ede6312af"]
    },
    "sendEmailNotification": true
}
```

In the example request, the `sendEmailNotification` is set to true.

Sample Response 1

Code

```
{
    "ucids": ["040ca427-21a2-4380-8b93-4c8e84a0721f"],
    "deletedProfiles": ["92ed24ede6312af"]
}
```

Sample Request 2 - (Using UCID)

Code

```
{
    "orgId": 1539,
    "region": "EU",
    "ids": {
        "ucid": ["040ca427-21a2-4380-8b93-4c8e84a0721f"]
    }
}
```

Sample Response 2

Code

```
{
    "ucids": ["040ca427-21a2-4380-8b93-4c8e84a0721f"],
    "deletedProfiles": ["040ca427-21a2-4380-8b93-4c8e84a0721f"]
}
```

Sample Request 3 - (Bulk Delete)

Code

```
{
  "orgId": 1539,
  "region": "EU",
  "ids": {
    "AdId": [
      "mock_adid_1",
      "mock_adid_2",
      "mock_adid_3",
      "mock_adid_4",
      "mock_adid_5"
    ]
  }
}
```

Sample Response 3

Code

```
{
    "ucids": [
        "3a4bc942-e709-4230-8fe8-300df3b5035c",
        "24da1c5d-a649-49e2-9392-e8ea9f2906f3",
        "84e051f0-aec3-4389-9c15-d6d61fe97d4f"
    ],
    "deletedProfiles": [
        "mock_adid_1",
        "mock_adid_2",
        "mock_adid_3"
    ],
    "profilesNotFound": [
        "mock_adid_4",
        "mock_adid_5"
    ]
}
```

Status Codes and Responses

The table below lists the different scenarios and their corresponding status codes and responses.

RESPONSE	STATUS/ERROR CODE	SCENARIO
"ucids": ["<Deleted UCID IDs>"] "deletedProfiles": ["<Set of Requested Ids which were deleted>"] "profilesNotFound": ["<Set Of Requested Ids which were not found> (if any)"]	200 OK	This is the response when your delete request is successful.
"error": "Cannot find any User Profiles with the provided IDs"	400 Bad Request	This is the response when you perform the deletion of an invalid or a deleted user.
"error": "Invalid user! Access denied."	401 Unauthorized	This is the response when either the user does not have access for the delete action or the access token is invalid.
"error": "Provided IDs matched with more than requested number of profiles! Deletion is only supported via Primary IDs"	400 Bad Request	This is the response when your delete request contains multiple profiles.
"error": "Incorrect region sent for your Organization!"	400 Bad Request	This is the response when you perform deletion with an invalid region code. For example, Region is passed as "AB" instead of "UK"/"EU".

RESPONSE	STATUS/ERROR CODE	SCENARIO
"error": "Limit Exceeded! We support only upto 400 profile deletions in a single request!"	400 Bad Request	This is the response when your request contains more than 400 unique IDs for deletion.
"error": "Cannot find any User Profiles with the provided IDs"	400 Bad Request	This is the response when your request contains the ID(s) that is already deleted.
"error": " User Profile is already deleted!"	400 Bad Request	This is the response when your request contains the UCID that is already deleted.
"error": "Keyspace: dmgr_1322 does not exist for region: EU"	400 Bad Request	This is the response when the Org Id in your request does not match with any profile.
"error": "Incorrect region sent for your Organization!"	400 Bad Request	This is the response when your request contains UK, but the region of the profile is EU.
"error": "orgId must not be null" "error": "ids must not be null" "error": "orgId must not be null"	400 Bad Request	This is the response when you misspell “orgId/region/ids”.
"error": "sendEmailNotification must not be null"	400 Bad Request	This is the response when the value for the key “sendEmailNotification” is missing.

RESPONSE	STATUS/ERROR CODE	SCENARIO
"error": "We support only ONE ID type in a single Delete request!"	400 Bad Request	This is the response when you perform deletion of two ID types.

Best Practices

For Delete API Best Practices and recommendations, refer here.

Next Topic → FAQs

Create or Update User Profiles
← Previous Page
How to Configure mTLS
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket

👍👎1.5. How to Configure mTLS

Main Topic ↑ Profile API

Perform the following steps to access Profile API using mTLS as the data transport method:

1. Zeotap provides a certificate and requests the SR (Certificate Signing Request) from you.

2. Sign the certificate and return it to Zeotap.

3. Zeotap's Cloudflare Manager will authorise the final certificate by signing it, and then return it to you.

4. Upload the final certificate in the settings or configuration section of your application.

5. Save or apply the changes. Afterward, you can access the APIs over mTLS via the endpoint https://mtls-api.zeotap.com.

Next Topic → Read API

Delete User Profiles
← Previous Page
FAQs
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket

1.6. FAQs

Main Topic ↑ Profile API

Frequently Asked Questions

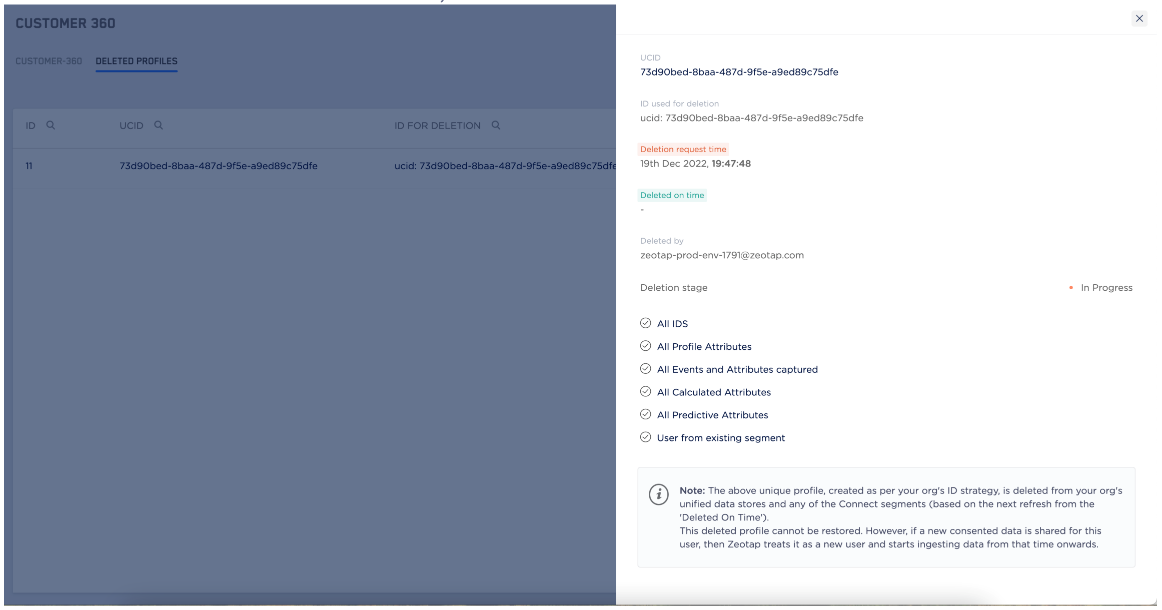
1. **How immediate is the deletion actioned? Can we assume a user profile to be fully deleted from Zeotap once the deletion HTTP request returns successfully?**

   A successful response is a confirmation of the receipt of the deletion request.

   When a delete request is received, the system initiates the process to delete the profile across the Zeotap system wherever this data is stored or referred. Subsequently, an email is triggered to the account Admin confirming the receipt of the deletion request with the details.

   It takes a couple of hours for the profile to be completely deleted across the Zeotap system. Note that this information is also mentioned in the notification email that is sent to the Admin.

To check the status, can refer to the Deleted profile log in your customer 360 interfaces. To know more refer, here.



ID	UCID	ID FOR DELETION
11	73d90bed-8baa-487d-9f5e-a9ed89c75dfe	ucid: 73d90bed-8baa-487d-9f5e-a9ed89c75dfe

UCID
73d90bed-8baa-487d-9f5e-a9ed89c75dfe

ID used for deletion
ucid: 73d90bed-8baa-487d-9f5e-a9ed89c75dfe

Deletion request time
19th Dec 2022, **19:47:48**

Deleted on time
-

Deleted by
zeotap-prod-env-1791@zeotap.com

Deletion stage • In Progress

- ✓ All IDS
- ✓ All Profile Attributes
- ✓ All Events and Attributes captured
- ✓ All Calculated Attributes
- ✓ All Predictive Attributes
- ✓ User from existing segment

Note: The above unique profile, created as per your org's ID strategy, is deleted from your org's unified data stores and any of the Connect segments (based on the next refresh from the 'Deleted On Time').
This deleted profile cannot be restored. However, if a new consented data is shared for this user, then Zeotap treats it as a new user and starts ingesting data from that time onwards.

2. **When I try to run the same deletion request a 2nd time (in other words, if I try to delete the same profile twice), I get a 400 HTTP response with the message "User Profile is already deleted". This suggests that Zeotap retains some information about the deleted users. Is this information eventually removed?**

As mentioned above, we maintain the log of the deleted profile. These are just for audit or reference purposes and are not connected to any activation system. These are eventually cleaned at the end of the next year, as per the GDPR specification.

How to Configure mTLS
← Previous Page
Next Page →

Need support?

Questions? Problems? Need more info? Contact us, and we can help!
Raise a ticket