Logistique Urbaine

For automation of Orders by OVO's Clients.

OVO's Orders API

INTRODUCTION

The Orders API provides an easy way to integrate your Orders data in Airtable with any external system. The API closely follows REST semantics, uses JSON to encode objects, and relies on standard HTTP codes to signal operation outcomes.

AIRTABLE

The API documentation below is specifically generated for your base. We recommend that you use the graphical Airtable interface to add a few records of example data for each table. These records will be displayed in the documentation examples generated below.

The ID of this base is `appkgeLOi6jsWevga`.
Please note: if you make changes to a field (column) name or type, the API interface for those fields will change correspondingly. Therefore, please make sure to update your API implementation accordingly whenever you make changes to your Airtable schema from the graphical interface.

Official API client:

- JavaScript: airtable.js (Node.js + browser)

Community-built API clients:

- Ruby: airrecord
- .NET: airtable.net

MATADATA

This API gives you the ability to list all of your bases, tables, fields, and views. You can register for access to the metadata API. For more, see the metadata API documentation.

RATE LIMITS

The API is limited to 5 requests per second per base. If you exceed this rate, you will receive a 429 status code and will need to wait 30 seconds before subsequent requests will succeed.

The official JavaScript client has built-in retry logic.

If you anticipate a higher read volume, we recommend using a caching proxy. This rate limit is the same for all plans and increased limits are not currently available.

AUTHENTICATION

Airtable uses simple token-based authentication. To generate or manage your API key, visit your account page. **Your API key carries the same privileges as your user account, so be sure to keep it secret!**

You can authenticate to the API by providing your API key in the HTTP authorization bearer token header. Alternatively, a slightly lower-security approach is to provide your API key with the `api_key` query parameter.
All API requests must be authenticated and made over HTTPS.

EXAMPLE USING BEARER TOKEN (RECOMMENDED)

```
$ curl https://api.airtable.com/v0/appkgeLOi6jsWevga/orders \
-H "Authorization: Bearer YOUR_API_KEY"
```

EXAMPLE USING QUERY PARAMETER

```
$ curl
https://api.airtable.com/v0/appkgeLOi6jsWevga/orders?api_key=YOUR_API_KEY
```

ORDERS TABLE

Fields

Each record in the orders table contains the following fields:

FIELD NAME
TYPE
DESCRIPTION

ID
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention`

Name
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Company
Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Address

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

City

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Zip code

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention`

From

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

To

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Phone

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Email

Long text

string

Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Notes
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Door code
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Floor
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Type
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Unit quantity
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Volume
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

Weight
Long text
string
Multiple lines of text, which may contain "mention tokens", e.g.
`<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

List orders records

To list records in orders, issue a **GET** request to the orders endpoint.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or `false`.
You can filter, sort, and format the results with the following query parameters. Note that these parameters need to be URL encoded. You can use our [API URL encoder tool](#) to help with this. If you are using a helper library like [Airtable.js](#), these parameters will be automatically encoded.

fields`array of strings`optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from ID and Name, send these two query parameters:

```
fields%5B%5D=ID&fields%5B%5D=Name
```

Note: `%5B%5D` may be omitted when specifying multiple fields, but must always be included when specifying only a single field.

filterByFormula`string`optional

A [formula](#) used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response.
If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.
For example, to only include records where ID isn't empty, pass in `NOT({ID} = '')` as a parameter like this:

```
filterByFormula=NOT%28%7BID%7D%20%3D%20%27%27%29
```

maxRecords`number`optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the Pagination section below for more.

pageSize`number`optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort`array of objects`optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.
The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.
For example, to sort records by ID in descending order, send these two query parameters:

```
sort%5B0%5D%5Bfield%5D=ID
sort%5B0%5D%5Bdirection%5D=desc
```

For example, to sort records by ID in descending order, pass in:

`[{field: "ID", direction: "desc"}]`

view`string`optional

The name or ID of a view in the orders table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat`string`optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.
`string`: cells will be formatted as user-facing strings, regardless of the field type. **Note:** You should not rely on the format of these strings, as it is subject to change.
The default is `json`.

timeZone`string`optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale`string`optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.
These parameters need to be URL encoded. If you are using a helper library like Airtable.js, they will be automatically encoded.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.
If there are more records, the response will contain an `offset`. To fetch the next page of records, include `offset` in the next request's parameters.
Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

```
curl
"https://api.airtable.com/v0/appkgeLOi6jsWevga/orders?maxRecords=3&view=Grid%
20view" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
{
    "records": [
        {
            "id": "recFChf9ewyL5X9Li",
            "fields": {
                "ID": "xx",
                "Name": "xx",
                "Company": "xx",
                "Address": "xx",
                "City": "xx",
                "Zip code": "xx",
                "Phone": "xx",
                "Email": "xx",
                "Notes": "xx",
                "Door code": "xx",
                "Floor": "xx",
                "Type": "xx",
                "Unit quantity": "xx",
                "Volume ": "xx",
                "Weight": "xx"
            },
            "createdTime": "2021-07-27T16:11:30.000Z"
        }
    ],
    "offset": "recFChf9ewyL5X9Li"
}
```

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Contact

For More information, please contact felipe@ovo.earth