Kargo Data Sharing API

Overview

The Kargo Data Sharing API service enables users to request reports based on pre-defined report specifications. The report can either be downloaded as a CSV file or fetched as a JSON response through the API.

Prerequisites

To access the API you will need to obtain user credentials from your Kargo account representative.

Accessing the API

The API can be accessed by passing requests to the base URL and appending required endpoints. The base URL for the API is:

`https://datasharingapi1.prod.kargocom/v1/`

Authenticating to the API

Users will initially authenticate by passing a base64 encoded string that consists of their username and password to the `authenticate` endpoint. The API will respond with a JSON result containing an API token to be used for the current session. Each session lasts for one hour.

Obtaining Your API Access Token

To obtain an API access token you must set `"Basic {YOUR BASE4 ENCODED USERNAME AND PASSWORD}"` as the value of a `POST` request header's `Authorization` key, like so:

`'Authorization: Basic your-base-64-encoded-email:password-string'`

Example:

```
curl --location --request POST 'https://datasharingapi.prod.kargocom/v1/authenticate' \
--header 'Authorization: Basic your-base-64-encoded-email:password-string'
```

Results:

```
{
   "token_type": "Bearer",
   "expires_in": 3600,
   "access_token": "123456abcdefgh09!"
}
```

Once you receive the token store it so that it is accessible for all your future API requests.

Creating A Report

POST

`/reports`

To create a report you must pass a POST request to the API's `report` endpoint. If the request is successful the response JSON will contain an id property. The returned id will be needed to check the status of the requested report.

The endpoint requires a `report_definition_id` whose value is the type of report to be returned along with the report definition's requested dimensions and measures. See this page for a listing of the type of report definitions available and their dimensions and measures.

Parameters

The table below describes the required and optional parameters that are to be sent with the JSON payload of the request:

Field	Type	Scope	Description
`report_definition_id`	String	Required	A string identifying which type of report to be included with the results.
`parameters`	Object	Required	Parent object containing the dimensions, measures, and filter objects for the request payload.
`parameters.dimensions`	Array	Required	A comma separated array of the dimensions to include in the request. See our full listing of dimensions for each report type.
`parameters.measures`	Array	Required	A comma separated array of the measures to be returned with the results. See our full listing of measures for each report type.
`parameters.filters`	Array	Optional	An array of objects to be used to filter the request results. See the Filters section below for more details.

Request Example

Click here to expand...

```
{
  "report_definition_id": "campaign-performance-metrics",
  "parameters": {
      "dimensions": [
          "campaign_name"
      ],
      "measures": [
          "clicks_sum"
      ],
      "filters": [
          {
              "day_time_id": {
                  "op": "between",
                  "start": 2023030100,
                  "end": 2023033100
              }
          }
      ]
  }
}
```

Results

Properties

If the request is successful Kargo will start processing the requested report. The API will respond with a JSON object containing an `id` propery. Store this value and use it when querying the report status. The table below lists all the properties returned from the request.

Property	Type	Definition
`id`	String	A unique id in string format that you will use to query the status of the requested report.
`report_definition_id`	String	The report definition id passed in the request.
`status`	String	The current status of the report generated from the request. Statuses are: • `pending` • `running` • `completed` • `error`
`created_at`	String	A date, in UTC format, when the report request was made.
`updated_at`	String	A date, in UTC format, when the report request was updated.
`rpt_file_expires_at`	String	A date in UTC format when the report request would expire.
`rpt_file_api_link`	String	A link that enables the user to view report data as a JSON response.
`rpt_file_download_link`	String	A url for downloading the report.
`record_count`	Integer	The number of results returned with the request.
`parameters`	Object	An object containing the dimensions, measures, and filter arrays sent with the request.
`parameters.dimensions`	Array	An array of the dimension values sent with the request. See our full

		listing of dimensions for each report type.
`parameters.measures`	Array	An array of the measure values sent with the request. See our full listing of measures for each report type.
`parameters.filters`	Array	An array of the filter values sent with the request. See the Filters section below for more details.

Response Example

Click here to expand...

```
{
    "id": "cgunomk22vhc70c9uaa0",
    "report_definition_id": "campaign-performance-metrics",
    "status": "pending",
    "created_at": "2023-04-17T17:05:30Z",
    "updated_at": "2023-04-17T17:05:30Z",
    "rpt_file_expires_at": null,
    "rpt_file_api_link": "",
    "rpt_file_download_link": "",
    "record_count": 0,
    "parameters": {
        "filters": [
        {
           "day_time_id": {
           "op": "between",
           "end": 2023030900,
           "start": 2023030900
        }
      }
    ],
    "measures": [
        "clicks_sum"
    ],
    "dimensions": [
        "campaign_name"
    ]
  }
}
```

Filters

Filters enable you to return specific data from a requested dimension rather than all data for that dimension. For example, if you wanted data for a specific time in a day you could apply the `day_time_id` filter to the request and select a start and end time for a date:

```
filters": [
    {
      "day_time_id": {
        "op": "between",
        "end": 2023030900,
        "start": 2023030900
      }
    }
  ]
```

⚠ At this time filters are only available for dimensions.

There are three syntax options for applying filters. You must use one of these formats:

Syntax 1

Submit a dimension ID with a supported value:

```
dimension_id: {
   "day_time_id": "2023100112"
}
```

Syntax 2

Submit a value operator along with the supported value:

```
"day_time_id": {
    "op": "gte",
    "start": 2023030900
}
```

Supported operators are:

Operator	Match Type	Description
`isNot`	Integer or Date	Will return items that are not the value.
`notIn`	[Integer] or [Date]	Will return items that are not in the list of values.
`neq`	Integer or Date	Not equal to the value.
`gt`	Integer or Date	Greater than the value.
`gte`	Integer or Date	Greater than or equal to the value.
`lt`	Integer or Date	Less than the value.
`lte`	Integer or Date	Less than or equal to the value.
`like`	String	Will return items that match a value. This operator is case sensitive.
`notLike`	String	Will return items that do not match a string value. This operator is case sensitive.
`iLike`	String	Will return items that match a value regardless of case. For example: `"campaign": {` `    "name": "Walmart"` `    "op": "iLike"` `}` Will return matches for `Walmart` and `walmart`.
`notILike`	String	Will return items that do not match a value regardless of

		case.

Syntax 3

Submit a range operator along with the request;

```
"day_time_id": {
    "op": "between",
    "end": 2023030900,
    "start": 2023030900
}
```

Supported operators are:

Operator	Description
`between`	The selected dimension value is between the start and end of the submitted range.
`notbetween`	The selected dimension value is not between the start and end of the submitted range.

Requesting Report Status

GET

`/reports?report_id={report-id}`

The API enables users to check the report request status. A best practice is to make requests in 15-second cadences until a status result of `complete` or `error` is returned.

If a report status indicates an error, check for errors in your request payload.

When a status is returned as `complete` you can request the completed report.

Arguments

Field	Type	Scope	Description
`report-id`	String	Required	The id returned in the initial report request.

Results

If the request is successful a JSON response will be returned with the following properties.

Property	Type	Description
`status`	String	The current status of the requested report. Values are: • `pending` • `running` • `completed` • `error`
`data`	Object	An object containing metadata about the requested report.

`data.total_rows`	Integer	The total number of items available from the request. Responses are limited to 10,000 row items.
`data.links`	Object	An object containing pagination links for the requested report.
`data.links.prev`	String	This link will return the previous range of items.
`data.links.next`	String	This link will return the next range of items.

```
{
  "status": "complete",
  "data": {
    "total_rows": 605,
    "links": {
      "prev": "",
      "next": "https://datasharingapi.dev.kargo.com/v1/completed-reports/cki35ccojls936p3cn9g?limit=3&offset=3"
    }
  }
}
```

Retrieving A Completed Report

Once a report has been created you can request to retrieve the report data in CSV or JSON format.

Retrieving CSV Format

GET

`v1/completed-reports/{report-id}/data`

This endpoint returns a link to a CSV file of the report data.

Parameters

Field	Type	Scope	Description
`report-id`	String	Required	The id returned in the initial report request.

Results

Returns a link to a CSV formatted file dependent on the dimensions and measures the user passed in their reporting request.

Property	Type	Description
`data`	Object	An object containing metadata about the requested report.
`data.total_rows`	Integer	The total number of items available from the request. Responses are limited to 10,000 row items.

`data.links`	Object	An object containing pagination links for the requested report.
`data.links.prev`	String	This link will return the previous range of items.
`data.links.next`	String	This link will return the next range of items.
`csv_link`	String	A link to the CSV file.

```
{
  "data": {
    "total_rows": 605,
    "links": {
      "prev": "",
      "next": "https://datasharingapi.dev.kargo.com/v1/completed-reports/cki35ccojls936p3cn9g?limit=3&offset=3"
    }
  },
  csv_link: "https://datasharingapi.dev.kargo.com/v1/completed-reports/cki35ccojls936p3cn9g/cki35ccojls936p3cn9g
}
```

Retrieving JSON Format

GET

`/v1/completed-reports/{report-id}?limit=200&offset=50`

This endpoint returns paginated report data in JSON format. By default, the API returns 100 records with the results. You can specify a limit amount as a query parameter. The maximum number of records that can be returned is 10,000.

If the total number of records is greater than the return limit you can paginate through the records by using the data.link.next property to view the next page.

Parameters

Field	Type	Scope	Description
`report-id`	String	Required	The id returned in the initial report request.

Query Parameters

Field	Type	Scope	Description
`limit`	Integer	Optional	Limits the number of items returned. Responses by default are limited to 100 items. The maximum limit is 10,000 items.
`offset`	Integer	Optional	The index of the first item to be returned. For example, if you set a `limit` of 500 and an `offset` of 250, the

			returned items would be index numbers 250-750.

Results

Property	Type	Description
`data`	Object	An object containing metadata about the requested report.
`data.total_rows`	Integer	The total number of items available from the request. Responses are limited to 10,000 row items.
`data.links`	Object	An object containing pagination links for the requested report.
`data.links.prev`	String	If a `limit` and `offset` are provided a link that will return the previous range of items.
`data.links.next`	String	If a `limit` and `offset` are provided a link that will return the next range of items.
`rows`	Array	An array of returned items.
`rows.Object`	Object	An object containing campaign data.
`rows.Object.campaign_name`	String	The name of the campaign associated with the data.
`rows.Object.media_delivery_sum`	Integer	The number of impressions delivered for the campaign.
`rows.Object.clicks_sum`	Integer	The number of clicks received by the ad.

```
{
  "data": {
    "total_rows": 605,
    "links": {
      "prev": "",
      "next": "https://datasharingapi.dev.kargo.com/v1/completed-reports/cki35ccojls936p3cn9g?limit=3&offset=3"
    },
    "rows": [
      {
        "campaign_name": "Chevron_ExtraMile Brand 2023",
        "media_delivery_sum": "2281",
        "clicks_sum": "5186"
      },
      {
        "campaign_name": "Austedo_TD DTC Unbranded 2023_ORION",
        "media_delivery_sum": "592",
        "clicks_sum": "149"
      },
      {
        "campaign_name": "AJR_Galveston Tourism_Q2Q3 2023",
        "media_delivery_sum": "230",
        "clicks_sum": "57"
      }
    ]
  }
}
```

Retrieving Available Report Definitions IDs

GET

`/v1/report-definitions/`

This endpoint returns a list of available report definition IDs.

Results

```
{
  "data": {
    "links": {
      "next": "string",
      "prev": "string"
    },
    "rows": [
      {}
    ],
    "total_rows": 0
  }
}
```

Errors

If the request is not successful the following HTTP status errors could be returned with the results:

HTTP Status Code	Definition
401	Authentication information is missing or invalid.
403	No access to the resource.
500	Server Error.

Python Example

You can use the Python example below as a reference for your request and handling the returned data.

Click here to expand...

```
import argparse
import base64
import http.client
import json
import logging
import math
import os
import ssl
import sys
```

```python
import time
from pprint import pprint

"""
This is an example script showing how to authenticate, request a report, waiting for report
to complete and viewing the data.  Since it's just a demo script, not a lot of care is given
to error checking ror does it try to account for all edge cases.

usage: data-sharing-api-example [-h] [--api_host API_HOST] --user USER [--password PASSWORD]

options:
  -h, --help           show this help message and exit
  --api_host API_HOST  Kargo Data Sharing API host
  --user USER          API user email
  --password PASSWORD  API user password; can be passed via env var, DATA_SHARING_API_USER_PASSWORD

e.g.
DATA_SHARING_API_USER_PASSWORD="password" python3 create_report.py --api_host=datasharingapi.prod.kargo.com
"""


status_error = 'error'

# Report payload example
report_payload = {
    'report_definition_id': 'campaign-performance-metrics',
    'parameters': {
        'dimensions': [
            'publisher_name',
            'advertiser_name',
            'campaign_name'
        ],
        'measures': [
            'advertiser_impressions_sum',
            'clicks_sum'
        ],
        'filters': [
            {
                'day_time_id': {
                'op': 'between',
                'start': 2023030900,
                'end': 2023030900
                }
            }
        ]
    }
}

def init_and_get_argparser():
    parser = argparse.ArgumentParser(prog='data-sharing-api-example')
    parser.add_argument('--api_host', help='Kargo Data Sharing API host', default='datasharingapi.prod.karg
    parser.add_argument('--user', help='API user email', required=True)
    # Note: passing clear text password via CLI is not good practice in production env.  Use env variable,
    default_api_user_password = os.getenv('DATA_SHARING_API_USER_PASSWORD')
    parser.add_argument('--password', help='API user password; can be passed via env var, DATA_SHARING_API_U
        required=(default_api_user_password == ''), default=default_api_user_password)
    return parser.parse_args()

```

```python
def init_and_get_logger(logger_id='data-sharing-api-report'):
    FORMAT = '%(asctime)s %(funcName)s %(levelname)s: %(message)s'
    logging.basicConfig(format=FORMAT)
    log = logging.getLogger(logger_id)
    log.setLevel(logging.DEBUG)
    return log

def base_request_header(access_token):
    return {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Authorization': f'Bearer {access_token}',
    }

def get_conn_response(conn, parse_json=True):
    res = conn.getresponse()
    data = res.read()
    if data is None:
        raise Exception('data is none')
    decoded_data = data.decode('utf-8')
    if parse_json:
        parsed_data = json.loads(decoded_data)
        return parsed_data
    return decoded_data

def authenticate(conn, username, password):
    tmp = f'{username}:{password}'.encode('ascii')
    base64_encoded_auth = base64.b64encode(tmp).decode('ascii')
    headers = {
        'Content-Type': 'application/json',
        'Authorization': f'Basic {base64_encoded_auth}'
    }
    conn.request('POST', '/v1/authenticate', None, headers)
    return get_conn_response(conn)

def create_report(conn, access_token, request_payload):
    payload = json.dumps(request_payload)
    headers = base_request_header(access_token)
    conn.request('POST', '/v1/reports', payload, headers)
    parsed_data = get_conn_response(conn)
    return parsed_data

def get_report_request(conn, access_token, report_id):
    headers = base_request_header(access_token)
    conn.request('GET', f'/v1/reports?show_all=1&report_id={report_id}', None, headers)
    parsed_data = get_conn_response(conn)
    return parsed_data.get('data', {})

def get_report_request_until_result(conn, access_token, report_id, log, timeout_seconds=180):
    """Attempt to get report request data."""
    sleep_sec = 10

    max_i = math.ceil(180 / sleep_sec)
    headers = base_request_header(access_token)
    i = 0
    while i <= max_i:
        data = get_report_request(conn, access_token, request_id)
        rows = data.get('rows', [])

        if len(rows) > 0:
            row = rows[0]
            if row.get('status') == status_error:
                log.warn('request encountered an error')
                break

            if row.get('rpt_file_api_link') != '':
                # View report data in JSON format.  Showing only the first page
                url = row.get('rpt_file_api_link') + '?limit=10'
                # If you want to download the file as CSV, you can use the following link instead
                # url = row.get('rpt_file_download_link')

                conn.request('GET', url, None, headers)
                data = get_conn_response(conn)
                print('Report Data:')
                pprint(data)
                break

        log.info(f'wait {i}...')
        time.sleep(sleep_sec)
        i = i + 1


if __name__ == '__main__':
    log = init_and_get_logger()
    args = init_and_get_argparser()
    log.debug(args)

    # Create HTTP connection instance
    conn = http.client.HTTPSConnection(args.api_host, context=ssl._create_unverified_context())

    # Authenticate user to get access token
    data = authenticate(conn, args.user, args.password)
    if data is None:
        log.error('no authentication data')
        sys.exit()
    access_token = data.get('access_token')
    if access_token == '':
        log.error('no access token')
        sys.exit()
    log.debug(f'access token: {access_token}')

    # Use access token to create a report
    data = create_report(conn, access_token, report_payload)
    request_id = data.get('id')
    if request_id is None:
        log.error('no report id, something went wrong')
        sys.exit()
    log.debug(f'report id: {request_id}')

    # Get report when it's processed or has error
    get_report_request_until_result(conn, access_token, request_id, log)
```

Send Feedback to:
ktc@kargo.com