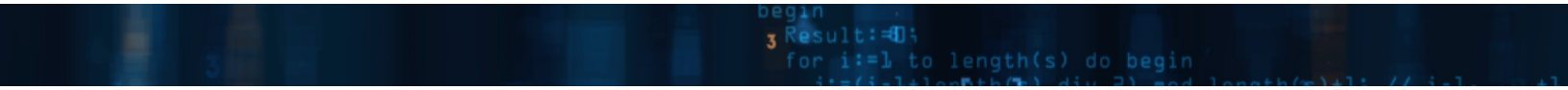
Quick Start

CADO API

cado//

Version: 2021-07-15 - Confidential



Contents

Overview

We provide an API for enabling 3rd party tools and automations scripts to integrate with the Cado Response platform. This document provides a quick onboarding that covers the main capabilities of the API and how you can best leverage them for your needs.

You can use the API to trigger operations in the platform (e.g. acquire instances from your cloud environments), retrieve data that our system captured (e.g. suspicious events), manage and control investigation projects, and more.

Readers of this manual

This manual assumes that the reader already has a basic understanding of Cado Response capabilities and will only guide on *how* to do things using the API. If you need a deeper explanation on the platform' capabilities, please follow our '*Cado Response User Manual*' beforehand.

Example Python Scripts

You can view end to end Python examples of the API on our GitHub repository.

API version

1.0.0

Getting Started

Introduction

The API is built on HTTPS and follows a RESTful architecture, which means it:

- Uses resource-oriented URLs
- Takes advantage of HTTP capabilities for passing parameters and authentication
- Responds with the standard HTTP response codes to indicate errors
- Returns JSON objects

To give you an idea of how to use the API, we have annotated the manual with samples.

Base URL

All API calls referenced in the manual start with a base URL which depends on the IP/Domain of where the platform was deployed, behind port 5000, for example:

Example

```
ec2example.compute.amazonaws.com:5000/api/v2
```

Date-Time Formats

The API uses Unix Timestamp as the default date/time format due the efficiency with indexing large amounts of data, sorting dated information and preventing time zones faults and mistakes that may occur when processing data from different systems.
We currently don't support other formats.

Status Codes

As mentioned above, the API uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the 2xx range indicate success. Codes in the 4xx range indicate an error that failed given the information provided and codes in the 5xx range indicate an error with Cado's servers.

The most used codes are:

CODE	DESCRIPTION
200	OK - Everything worked as expected
400	Bad Request - Usually occur due incorrect/missing parameters
401	Not Authorized - None or an invalid token was provided
403	Forbidden - A valid token was provided but the user don't have permissions
404	Not Found - The resource (URL) not exists
405	Method Not Allowed - The resource exists but don't support the given http method
409	Conflict - Usually occur due existing values (e.g. username exists)
5xx	Server Errors - If it's an ongoing issue, please contact Cado for support

For the full list Click Here

Request Headers

Any request to the API needs to include the following header:

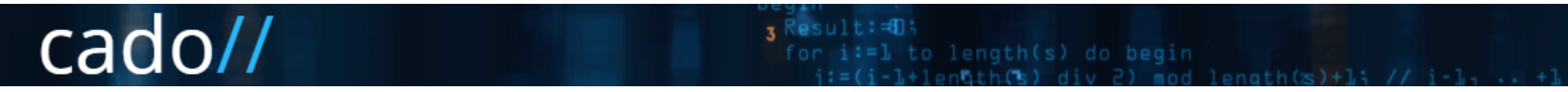
Authentication Header

```
Authorization: Bearer <API Key>
```

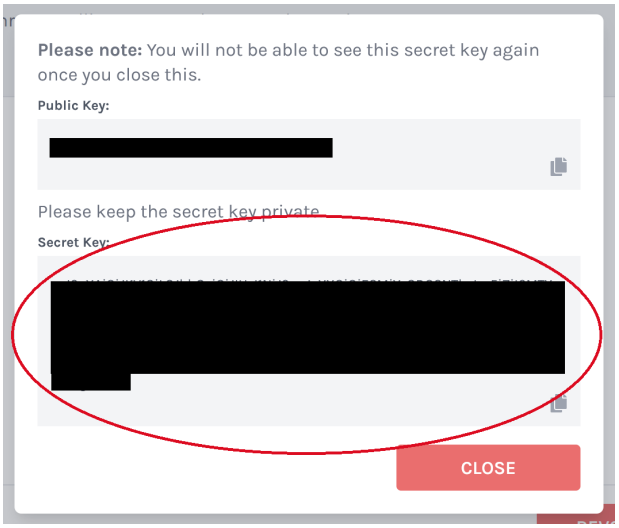
In additional to the authentication header discussed above, for any request, unless stated otherwise, the following headers must be provided:

Headers

```
Content-Type: application/json
```



Authentication



Before diving into any of the workflows we can perform and the functionality provided by the API, we firstly need to understand what is the correct way we should access it.

To be able to access the different resources of the API, any request must be provided with a valid API Key generated by our authentication server.
To generate that key, go to the "*Settings*" page (bottom left), then choose the "*API*" tab and click on the *"Create New Key"* button. You should see a popup with a secret and public key*.
The secret key is the API Key that we need to provide with each request, and the public key is the identifier that we can use for revoking the key later on if needed.

---

* You will not be able to see the secret key again once you close the pop up, so make sure to keep it in a secure place.

Examples

Notes:

- All the URLs in the examples are relative to the base url
- Placeholders marked with the color green need to be replaced with real values

Create a new project

Create a new project (return project_id to be used in other examples)

```
POST /projects
Authorization: Bearer <API Key>
Content-Type: application/json

{
    "caseName": "projectX"
}
```

Import EC2 Instance to a project

1. List all ec2 instances (So we can see the instances' ids)

```
GET /projects/<project_id>/imports/ec2
Authorization: Bearer <API Key>
Content-Type: application/json
```

2. Acquire an instance as a new evidence in our project

```
POST /projects/<project_id>/imports/ec2
Authorization: Bearer <API Key>
Content-Type: application/json

{
    "instance_id": "i-xxxxxxxxxxxxx",
    "bucket": "test-bucket
}
```

Import from S3

1. List buckets (So we can see the files)

```
GET /projects/<project_id>/imports/s3?bucket=<some_bucket_name>
Authorization: Bearer <API Key>
Content-Type: application/json
```

2. Acquire a file from a bucket as a new evidence in our project

```
POST /projects/<project_id>/imports/s3
Authorization: Bearer <API Key>
Content-Type: application/json

{
    "bucket": "some_name",
    "file_name": "folderA/folder/some_file.dd"
}
```

Processing Progress

Processing evidence can take a while and often you'd like to get the progress of that process. Each successful import request will return pipeline_id that can be used as following:

Progress of processing

```
GET /tasks/pipelines?pipeline_id=<pipeline_id>
Authorization: Bearer <API Key>
Content-Type: application/json
```

Querying the timeline

Great! You successfully processed evidence using the API, now you want to get some data!
The Timeline resource can get many different filters as url attribute, here is some of the options:

- **from_timestamp** int, unix timestamp - Show results that occur after this time
- **to_timestamp** int, unix timestamp - Show results that occur before this time
- **severity** int, event severity range - Events with severity between 1 to the given value
- **evidence_id** int, evidence unique id - Show events from a specific evidence
- **query** str, search across fields - Show results that the given string appears in any field
- **page** int, what page to show default: 1
- **per_page** int, how many results to show per page default: 100

Querying the timeline example

```
GET /projects/<project_id>/timeline?query=can_be_anything
Authorization: Bearer <API Key>
Content-Type: application/json
```