

```
api.txt x
# API Definition
## Entity Definitions
### Charge:
- id: uuid
- customer_id: uuid
- amount: integer
- currency: string (or currency-code enum)
- status: enum ["succeeded", "pending", "failed"]
### Customer:
- id: uuid
- name: string
- address: string
- email: string
- card: Card
### Card
## Endpoint Definitions
### Charges
CreateCharge(charge: Charge)
  => Charge
GetCharge(id: uuid)
  => Charge
UpdateCharge(id: uuid, updatedCharge: Charge)
  => Charge
ListCharges(offset: integer, limit: integer)
  => Charge[]
CaptureCharge(id: uuid)
  => Charge
### Customers
CreateCustomer(customer: Customer)
  => Customer
GetCustomer(id: uuid)
  => Customer
UpdateCustomer(id: uuid, updatedCustomer: Customer)
  => Customer
DeleteCustomer(id: uuid)
  => Customer
ListCustomers(offset: integer, limit: integer)
  => Customer[]

charge.json x
{} charge.swagger.json
{} customer.json x
{} api-swagger.yaml

{} charge.json > ...
1 {
2   "id": "a92b1cd0-0844-4f3f-badb-a15a1dd0f4d5",
3   "customer_id": "66b89078-2516-4659-adee-3fa6f453089b",
4   "amount": 10000,
5   "currency": "usd",
6   "status": "pending"
7 }
8

{} customer.json > ...
1 {
2   "id": "66b89078-2516-4659-adee-3fa6f453089b",
3   "name": "Cersei Lannister",
4   "address": "1 King's Landing",
5   "email": "cersei.lannister@gmail.com",
6   "card": {}
7 }
8
```

```
api.txt x
# API Definition
## Entity Definitions
### Charge:
- id: uuid
- customer_id: uuid
- amount: integer
- currency: string (or currency-code enum)
- status: enum ["succeeded", "pending", "failed"]
### Customer:
- id: uuid
- name: string
- address: string
- email: string
- card: Card
### Card
## Endpoint Definitions
### Charges
CreateCharge(charge: Charge)
  => Charge
GetCharge(id: uuid)
  => Charge
UpdateCharge(id: uuid, updatedCharge: Charge)
  => Charge
ListCharges(offset: integer, limit: integer)
  => Charge[]
CaptureCharge(id: uuid)
  => Charge
### Customers
CreateCustomer(customer: Customer)
  => Customer
GetCustomer(id: uuid)
  => Customer
UpdateCustomer(id: uuid, updatedCustomer: Customer)
  => Customer
DeleteCustomer(id: uuid)
  => Customer
ListCustomers(offset: integer, limit: integer)
  => Customer[]

charge.json x
{} charge.swagger.json x
{} customer.json x
{} api-swagger.yaml x

{} charge.json > ...
1 {
2   "swagger": "2.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "Example Stripe API"
6   },
7   "host": "api.stripe.com",
8   "basePath": "/v1",
9   "schemes": [ "http", "https" ],
10  "consumes": [ "application/json" ],
11  "produces": [ "application/json" ],
12  "paths": {
13    "/v1/charges": {
14      "get": {
15        "summary": "List all charges",
16        "operationId": "listCharges",
17        "parameters": [
18          {
19            "name": "offset",
20            "in": "query",
21            "description": "How many items to skip in result set",
22            "required": false,
23            "type": "integer",
24            "format": "int32"
25          },
26          {
27            "name": "limit",
28            "in": "query",
29            "description": "How many items to return at once",
30            "required": false,
31            "type": "integer",
32            "format": "int32"
33          }
34        ],
35        "responses": {
36          "200": {
37            "description": "A paginated array of charges",
38            "schema": {
39              "$ref": "#/definitions/Charges"
40            }
41          }
42        }
43      }
44    }
45  }
46 }

{} customer.json > ...
1 {
2   "swagger": "2.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "Example Stripe API",
6     "host": "api.stripe.com",
7     "basePath": "/v1",
8     "schemes": [ "http", "https" ],
9     "consumes": [ "application/json" ],
10    "produces": [ "application/json" ],
11    "paths": {
12      "/v1/charges": {
13        "get": {
14          "summary": "List all charges",
15          "operationId": "listCharges",
16          "parameters": [
17            {
18              "name": "offset",
19              "in": "query",
20              "description": "How many items to skip in result set",
21              "required": false,
22              "type": "integer",
23              "format": "int32"
24            },
25            {
26              "name": "limit",
27              "in": "query",
28              "description": "How many items to return at once",
29              "required": false,
30              "type": "integer",
31              "format": "int32"
32            }
33          ],
34          "responses": {
35            "200": {
36              "description": "A paginated array of charges",
37              "schema": {
38                "$ref": "#/definitions/Charges"
39              }
40            },
41            "201": {
42              "description": "Null response"
43            }
44          }
45        }
46      }
47    }
48  }
49 }

{} api-swagger.yaml > ...
1 swagger: "2.0"
2 info:
3   version: 1.0.0
4   title: Example Stripe API
5   host: api.stripe.com
6   basePath: /v1
7 schemes:
8   - http
9   - https
10 consumes:
11   - application/json
12 produces:
13   - application/json
14 paths:
15   /v1/charges:
16     get:
17       summary: List all charges
18       operationId: listCharges
19       parameters:
20         - name: offset
21           in: query
22           description: How many items to skip in result set
23           required: false
24           type: integer
25           format: int32
26         - name: limit
27           in: query
28           description: How many items to return at once
29           required: false
30           type: integer
31           format: int32
32       responses:
33         "200":
34           description: A paginated array of charges
35           schema:
36             $ref: '#/definitions/Charges'
37         "201":
38           description: Null response
39       post:
40         summary: Create a charge
41         operationId: createCharge
42         responses:
43           "201":
44             description: Null response
45           "200":
46             description: A charge object
47             schema:
48               $ref: '#/definitions/Charge'
```

Stripe API Reference

Find anything

Introduction

Authentication

Connected Accounts

Errors

Expanding Responses

Idempotent Requests

Metadata

Pagination

Request IDs

Versioning

CORE RESOURCES

Balance

Balance Transactions

Charges

Customers

Disputes

Events

Files

File Links

Mandates

PaymentIntents

SetupIntents

Payouts

Products

Prices

Refunds

Tokens

Sign In

API Reference

The Stripe API is organized around **REST**. Our API has predictable resource-oriented URLs, accepts **form-encoded** request bodies, returns **JSON-encoded** responses, and uses standard HTTP response codes, authentication, and verbs.

You can use the Stripe API in **test mode**, which does not affect your live data or interact with the banking networks. The API key you use to **authenticate** the request determines whether the request is live mode or test mode.

The Stripe API differs for every account as we release new **versions** and tailor functionality. Log in to see docs customized to your version of the API, with your test key and data.

Subscribe to Stripe's API announce mailing list for updates.

Was this section helpful? Yes No

JUST GETTING STARTED?

Check out our [development quickstart guide](#).

NOT A DEVELOPER?

Use apps from our [partners](#) to get started with Stripe and to do more with your Stripe account—no code required.

BASE URL

<https://api.stripe.com>

CLIENT LIBRARIES

Ruby Python PHP Java Node.js Go .NET

By default, the Stripe API Docs demonstrate using **curl** to interact with the API over HTTP. Select one of our official [client libraries](#) to see examples in code.

Authentication

20:34 28:35

Sign In

algexpert.io

Google Cloud

Cloud IoT Core

REST reference

API overview

cloudiot

Overview

v1

REST Resources

projects.locations.registries

projects.locations.registries...

Overview

create

delete

get

list

modifyCloudToDeviceCo...

patch

sendCommandToDevice

projects.locations.registries...

projects.locations.registries...

Types

BindDeviceToGatewayResp...

GatewayListOptions

ListDeviceConfigVersionsRe...

ListDeviceStatesResponse

ListDevicesResponse

LogLevel

Policy

SendCommandToDeviceRe...

TestIamPermissionsRespon...

UnbindDeviceFromGateway...

cloudiotdevice

gcloud CLI

Client library for Android Things

SDK for embedded C/C++

Cloud IoT Core > Documentation > Reference

REST Resource: projects.locations.registries.devices

Send feedback

Resource: Device

The device resource.

JSON representation

```
{
  "id": string,
  "name": string,
  "numId": string,
  "credentials": [
    {
      object(DeviceCredential)
    }
  ],
  "lastHeartbeatTime": string,
  "lastEventTime": string,
  "lastStateTime": string,
  "lastConfigAckTime": string,
  "lastConfigSendTime": string,
  "blocked": boolean,
  "lastErrorTime": string,
  "lastErrorStatus": {
    object(Status)
  },
  "config": {
    object(DeviceConfig)
  },
  "state": {
    object(DeviceState)
  }
}
```

Contents

Resource: Device

DeviceCredential

PublicKeyCredential

PublicKeyFormat

Status

GatewayConfig

GatewayType

GatewayAuthMethod

Methods

create

delete

get

list

modifyCloudToDeviceConfig

patch

sendCommandToDevice

algexpert.io

The screenshot shows the Twitter Developer API documentation page for the `get-statuses-home_timeline` endpoint. The sidebar on the left contains navigation links for 'Documentation', 'Twitter API', 'Getting started', 'Tutorials', 'Tools and libraries', 'Migrate', 'API reference Index', 'Twitter API v2' (with an 'Early Access' badge), and 'Twitter API v1.1'. The main content area displays the endpoint details, including a table of parameters.

Name	Required	Description	Default Value	Example
count	optional	Specifies the number of records to retrieve. Must be less than or equal to 200. Defaults to 20. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied.		5
since_id	optional	Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available.		12345
max_id	optional	Returns results with an ID less than (that is, older than) or equal to the specified ID.		54321
trim_user	optional	When set to either <code>true</code> , <code>1</code> or <code>1</code> , each Tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object.		<code>true</code>
exclude_replies	optional	This parameter will prevent replies from appearing in the returned timeline. Using <code>exclude_replies</code> with the <code>count</code> parameter will mean you will receive up-to count Tweets — this is because the <code>count</code> parameter retrieves that many Tweets before filtering out retweets and replies.		<code>true</code>
include_entities	optional	The <code>entities</code> node will not be included when set to <code>false</code> .		<code>false</code>

4 Prerequisites

HTTP

The **HyperText Transfer Protocol** is a very common network protocol implemented on top of TCP. Clients make HTTP requests, and servers respond with a response.

Requests typically have the following schema:

```
host: string (example: algoexpert.io)
port: integer (example: 80 or 443)
method: string (example: GET, PUT, POST, DELETE, OPTIONS or PATCH)
headers: pair list (example: "Content-Type" => "application/json")
body: opaque sequence of bytes
```

Responses typically have the following schema:

```
status code: integer (example: 200, 401)
headers: pair list (example: "Content-Length" => 1238)
body: opaque sequence of bytes
```

JSON

A file format heavily used in APIs and configuration. Stands for **JavaScript Object Notation**. Example:

```
{
  "version": 1.0,
  "name": "AlgoExpert Configuration"
}
```

YAML

A file format mostly used in configuration. Example:

```
version: 1.0
name: AlgoExpert Configuration
```

ACL

Short for **Access-Control List**. This term is often used to refer to a permissioning model: which users in a system can perform which operations. For instance, APIs often come with ACLs defining which users can delete, edit, or view certain entities.

2 Key Terms



| Pagination

When a network request potentially warrants a really large response, the relevant API might be designed to return only a single **page** of that response (i.e., a limited portion of the response), accompanied by an identifier or token for the client to request the next page if desired.

Pagination is often used when designing **List** endpoints. For instance, an endpoint to list videos on the YouTube Trending page could return a huge list of videos. This wouldn't perform very well on mobile devices due to the lower network speeds and simply wouldn't be optimal, since most users will only ever scroll through the first ten or twenty videos. So, the API could be designed to respond with only the first few videos of that list; in this case, we would say that the API response is **paginated**.

| CRUD Operations

Stands for **Create, Read, Update, Delete** Operations. These four operations often serve as the bedrock of a functioning system and therefore find themselves at the core of many APIs. The term **CRUD** is very likely to come up during an API-design interview.