Advanced Trade API Legacy Key Authentication

You can create and activate new API keys in your API settings. Please note, legacy API key authentication is supported for both Advanced Trade and Coinbase App APIs for existing keys. You can no longer create new keys using legacy process.

Legacy API keys are for individuals or applications for individual use. They can be used with Advanced REST APIs & WebSocket channels (except new features).

Signing Requests

All Advanced Trade REST API requests with legacy API keys must contain the following headers:

Header	Description
`CB-ACCESS-KEY`	API key as a string (that you create on coinbase.com)
`CB-ACCESS-SIGN`	Encoded signature using API secret
`CB-ACCESS-TIMESTAMP`	Timestamp for your request

Tip

Advanced Trade **does not** require a `PASSPHRASE` as did Coinbase Pro.

Creating a Signature

1. Create a signature string by concatenating the values of these query parameters with the `+` operator: `timestamp + method + requestPath + body`.
   - `timestamp` is the same as the `CB-ACCESS-TIMESTAMP` header (+/-30 seconds)
   - `method` should be UPPER CASE
   - `requestPath` is the full path (minus the base URL and query parameters), for example:
     - `/api/v3/brokerage/orders/historical/fills`
     - `/api/v3/brokerage/products/BTC-USD/ticker`
   - `body` is the request body string -- it is omitted if there is no request body (typically for `GET` requests)
2. Create a sha256 HMAC object with your API secret on the signature string.
3. Get the hexadecimal string representation of the sha256 HMAC object and pass that in as the `CB-ACCESS-SIGN` header.

Lowercase signature

A signature must be in lowercase letters or the program throws a 401 error.

Signature Code Samples

The following examples demonstrate how to generate a signature in Python, Ruby, and JavaScript:

Python Ruby JavaScript

```
import json, hmac, hashlib, time, base64
#timestamp = str(int(time.time()))
#request.method = GET or POST
#request.path_url.split('?')[0] = /api/v3/brokerage/orders/historical/batch
message = timestamp + request.method + request.path_url.split('?')[0] +
str(request.body or '')
signature = hmac.new(secretKey.encode('utf-8'), message.encode('utf-8'),
digestmod=hashlib.sha256).digest()
print(signature.hex(), ts)
```

Tip

The Advanced Trade `requestPath` should only include the path of the API endpoint in the string for hashing. It should *not* include the base URL (protocol and domain) *nor* any query parameters. By contrast, the Coinbase App `requestPath` does include query parameters.

API	`requestPath`	Valid Example
Advanced (v3)	API endpoint	`/api/v3/brokerage/products/BTC-USD/ticker`
Coinbase App (v2)	API endpoint + query params	`/v2/exchange-rates?currency=USD`

Making Requests

All private API requests must include CB-ACCESS-* headers:

1. Set a timestamp for the `CB-ACCESS-TIMESTAMP` header.
2. Create an encoded signature as the `CB-ACCESS-SIGN` header (with the API secret).
3. Set your legacy API key for the `CB-ACCESS-KEY` header.
4. Apply the headers to the request. You are ready to send.

Example Request

```
curl https://api.coinbase.com/v3/brokerage/accounts \
  --header "CB-ACCESS-KEY: <your api key>" \
  --header "CB-ACCESS-SIGN: <the user generated message signature>" \
  --header "CB-ACCESS-TIMESTAMP: <a timestamp for your request>"
```

All requests should have content type `application/json` and the body must be valid JSON.

```
# Ruby code sample of a GET Request to product ticker

require 'uri'
require 'net/http'
require 'openssl'

url = URI("https://coinbase.com/api/v3/brokerage/products/BTC-USD/ticker?limit=3")
request_path= "/api/v3/brokerage/products/BTC-USD/ticker"
body = ""
method = "GET"

timestamp = Time.now.to_i
payload = "#{timestamp}#{method}#{request_path}#{body}"
# create a sha256 hmac with the secret
signature = OpenSSL::HMAC.hexdigest('sha256', $SECRET_KEY, payload)

http = Net::HTTP.new(url.host, url.port)
http.use_ssl = true
http.use_ssl = true

request = Net::HTTP::Get.new(url)
request["accept"] = 'application/json'
request["CB-ACCESS-KEY"] = $ACCESS_KEY
request["CB-ACCESS-SIGN"] = signature
request["CB-ACCESS-TIMESTAMP"] = timestamp

response = http.request(request)
puts response.read_body
```

*Last updated on **Sep 12, 2024***

Need help? Ask in our Discord