

Historical Time Series Web Service

To gain access to the Historical Time Series Web Service please contact:



Americas

Group Sales Line

+1-212-723-3811

citivelocitysales@citi.com



Europe

Group Sales Line

+44-20-3569-4308

emea.citivelocity@citi.com



Asia Pacific

Group Sales Line

+852-2501-2682

ap.velocity@citi.com

Data pulled from this web service
cannot be redistributed!

Overview

The purpose of this web service is to enable clients of Citi Velocity to programmatically access the Historical Time Series data which powers Citi Velocity Charting.

Limitations

This web service employs a stateless request/response mechanism. It cannot be used to frequently poll for the latest updates for the same series, and measures are in place to prevent this (more details below).

Q: Does this mean I cannot make frequent requests to the service?

A: Frequent requests are fine (exactly how frequent is detailed below). You simply cannot make frequent requests *for the same series*.


Q: Can I poll infrequently, say several (<10) times per day, to fetch the latest values for the same series?

A: Yes.

Q: Since I can only poll infrequently for the same series, how do I know when to make my requests?

A: We provide timing information on when data usually becomes available for a given series via a Metadata API. Details below.

The following types of time series are not supported by this service:

- Math functions
- Portfolio series
- Series which are not exportable (marked with a lock icon in the Data Browser: .

Q: Why no math functions? Can support be added for math functions?

A: This service only provides the raw time series data. No post-processing of the data will be supported. This includes math functions, interpolation, technical analyses, and other features available on the website.

Historical

The word “historical” is used here in contrast to “live”. This should not be confused with EOD vs intraday. For example, this service does provide historical intraday data down to a minutely frequency for some series. However, due to the limitations discussed above, the minutely data cannot be polled for updates on an ongoing minute-by-minute basis (or even hour-by-hour) – so it is not available live through this service.

There is a separate Live service which utilizes Websockets to push minutely updates. Please see the [Live API](#) document for details.

Time Series

Charting has other types of data, such as curve data, but this web service only provides access to time series data.

Q: Will curve data be made available at some point?


A: There are no plans for this, but it is possible if there is enough demand for it.

Initial Setup

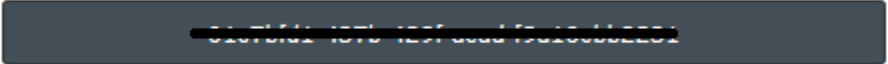
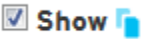
New Users

Please see the [API User Guide](#) for details on how to setup your credentials. In brief:

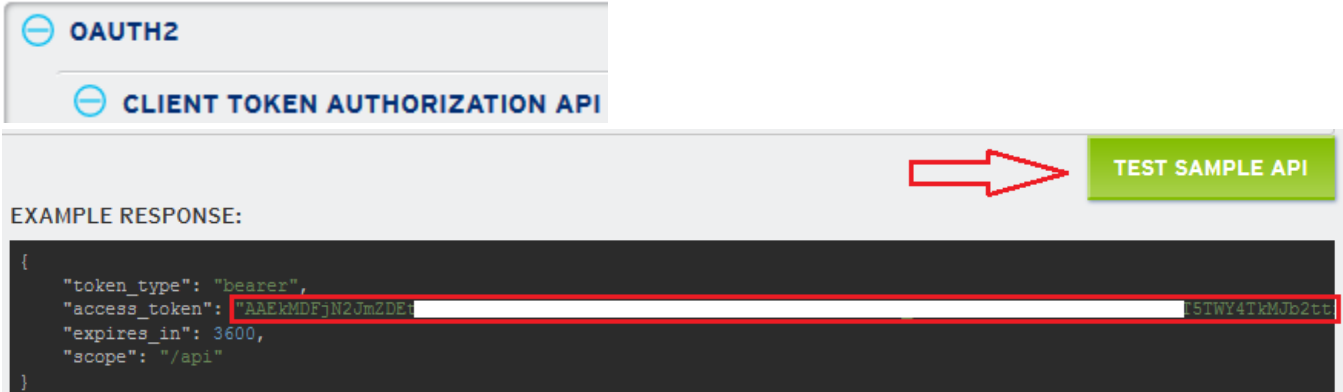
1. Navigate to [Market Buzz > Citi Velocity API Marketplace](#) on the Citi Velocity website.


2. Click 


3. Copy out your Client Id:


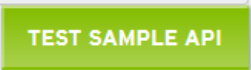
Client Id:  

4. Generate and copy out your access token:



 OAUTH2

 CLIENT TOKEN AUTHORIZATION API

EXAMPLE RESPONSE:

```
{
  "token_type": "bearer",
  "access_token": "AAEkMDFjN2JmZDE4LTU0YTY4TlRlY2E4",
  "expires_in": 3600,
  "scope": "/api"
}
```

Note that this token expires after one hour, and can also be fetched programmatically (rather than by clicking “Test Sample API”). See the API User Guide for details.

Legacy User Migration

Legacy users – those who are using the “login” and “passcode” query parameters to authenticate – will need to migrate to the new APIm approach described above.

Legacy		Replacement
“login” query parameter	➔	“client_id” query parameter
“passcode” query parameter	➔	“authorization” <u>http header</u>
www.citivelocity.com/analytics/eppublic/charting/ data /v1		
➔	api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/ data	
www.citivelocity.com/analytics/eppublic/charting/ <xyz> /v1		
➔	api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/ <xyz>	

Data API

<https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/data>

Request & Response Format By Example

The request and response are both serialized using JSON. HTTP method for requests is POST.

Example 1 – Request

Here is an example of the simplest possible request, which fetches one week of Gold prices:

```
{
  "startDate": 20170108,
  "endDate": 20170114,
  "tags": [
    "COMMODITIES.SPOT.SPOT_GOLD"
  ]
}
```

tags

An array of between 1 and 100 distinct tags. The tag structure can be explored using the Data Browser in Charting

Q: Can I get a listing of all tags?

A: No, but we do provide a tag listing API – see below for details. We are also happy to answer specific questions about the tag structure.

Example 2 – Request

Here is an example of how to pull one day's worth of hourly intraday data:

```
{
  "startDate": 20170323,
  "endDate": 20170323,
  "tags": [
    "FX.SPOT.EUR.USD.CITI"
  ],
  "startTime": 0,
  "endTime": 2359,
  "frequency": "HOURLY"
}
```

startTime & endTime

Format is hhmm – two digit (24) hour × 100 + two digit minute. There is no need to prepend leading zeros. start/endTime is combined with start/endDate to form a date-time. All times are in GMT.

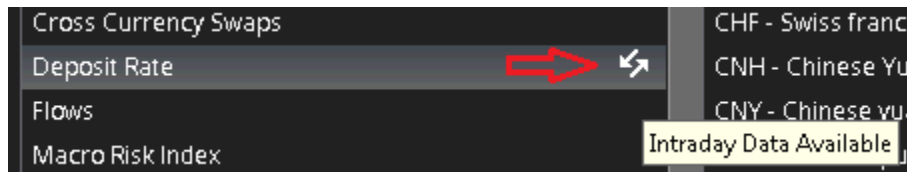
frequency

Value	Description	Max History
MONTHLY	Monthly	-
WEEKLY	Weekly	-
DAILY	Daily - this is the default	-
HOURLY	Hourly	1 Year
MI10	Ten-minutely	2 Months
MI01	Minutely	1 Month

HOURLY and finer are the intraday frequencies. DAILY and coarser are the EOD frequencies.

The intraday frequencies have a limit on the amount of history which can be pulled in a single request via this service. There is a separate service for bulk exporting intraday data. Access to the bulk export API is granted separately – please contact your Citi representative for details.

All series have EOD data, but only a relatively small number of series have intraday data. Series with intraday data are marked with an icon in the Data Browser:



The highlights for intraday data are:

Category	Sub Categories
Equities	Stocks, Indices, ETF
FX	Spot, Forward, Deposit Rate, Vol (Implied)
Rates	Sovereign, Par Swap

See the [Intraday Tags](#) spreadsheet for a complete listing.

We are working to onboard additional intraday data.

Example 4 – Request

Here is an example of how to pull OHLC data:

```
{
  "startDate": 20170320,
  "endDate": 20170324,
  "tags": [
    "FX.SPOT.EUR.USD.CITI"
  ],
  "pricePoints": "OHLC"
}
```

pricePoints

Value	Description
C	Closes only – this is the default
OHLC	Open, High, Low, and Close

Other subsets of OHLC such as “High and Low only” are not supported. Open, High, and Low are typically only available for those series which support intraday frequencies. However, they are available for all series at the Weekly and Monthly frequencies (for which they can be approximated using the Daily values).

latestOnly

Optional boolean value. Only the last point will be returned for each series. Useful for reducing the size of the response when only the latest value is of interest. startDate and endDate are still required in the request.

Example 1 – Response

```
{
  "frequency": "DAILY",
  "body": {
    "COMMODITIES.SPOT.SPOT_GOLD": {
      "x": [
        20170109, 20170110, 20170111, 20170112, 20170113
      ],
      "c": [
        1178.5, 1189.5, 1178.55, 1205.05, 1190.35
      ],
      "type": "SERIES"
    }
  },
  "status": "OK"
}
```

status

Possible values are “OK” and “ERROR”. An ERROR response will have a “message” field, e.g.

```
{"message": "startDate cannot be after endDate", "status": "ERROR"}
```

An OK response will have the “body” and “frequency” fields.

Response Frequency

The frequency is present in the response because it can be coarser than the requested frequency. This occurs in two scenarios:

1. Too much history is requested for an intraday frequency. For example, if two weeks of Minutely data is requested then two weeks of Ten-Minutely data will be returned instead. If two years of Minutely data is requested then two years of Daily data will be returned.
2. Intraday data is requested for a series that only has EOD data. The frequency will be bumped up to Daily.

body

The body is a map from requested tags to per-tag response objects, which contain the following fields:

type

Possible values are “SERIES” and “ERROR”. An ERROR response will have a “message” field, e.g.

```
{"message": "Bad tag: FOO.BAR.BAD.TAG", "type": "ERROR"}
```

Note: the response frequency will be null if all per-tag responses have type ERROR.

A SERIES response will have the “x” and “c” fields.

x & c

Two arrays of equal length. Both arrays will not contain any null values. “c” stands for Close – these are the y-values. The “x” array contains the dates or date-times in the following frequency-specific format:

Frequency	Format
Monthly	yyyyMM
Weekly	yyyyww
Daily	yyyyMMdd
Hourly	yyyyMMddHH
Ten-Minutely	yyyyMMddHHm
Minutely	yyyyMMddHHmm

Date Field	Description
yyyy	4 digit: year (1980, 2017, etc.)
MM	2 digit: month, 1 - 12
yyyy	4 digit: ISO week-based year
ww	2 digit: ISO week of week-based year
dd	2 digit: date, 1 - 31
HH	2 digit: hour, 0 - 23
m	1 digit: ten-minute value, 0 - 5
mm	2 digit: minute, 0 - 59

All times are in GMT.

Q: Why is there no day for the Monthly frequency, no minute for Hourly, etc.?

A: Let’s use Monthly as the representative example. What day should it be? If there is only one point during a month, we could think about using the day from that point, but what if there’s more than one point? Take the last day for Close? What about for OHLC?

Note: the “x” & “c” arrays can be empty if the tag is valid, but has no data associated with it (either at all, or no data during the requested time period). This is not treated as an error.

Example 2 – Response

```
{
  "frequency": "HOURLY",
  "body": {
    "FX.SPOT.EUR.USD.CITI": {
      "x": [
        2017032300, 2017032301, 2017032302, 2017032303, 2017032304, 2017032305,
        2017032306, 2017032307, 2017032308, 2017032309, 2017032310, 2017032311,
        2017032312, 2017032313, 2017032314, 2017032315, 2017032316, 2017032317,
        2017032318, 2017032319, 2017032320, 2017032321, 2017032322, 2017032323
      ],
      "c": [
        1.0786, 1.0786, 1.0785, 1.079, 1.0793, 1.0791,
        1.08, 1.0795, 1.0775, 1.0771, 1.0788, 1.0785,
        1.0787, 1.0785, 1.0789, 1.0787, 1.0788, 1.0782,
        1.0784, 1.0784, 1.0783, 1.0784, 1.0782, 1.0783
      ],
      "type": "SERIES"
    }
  },
  "status": "OK"
}
```

Example 3 – Response

```
{
  "frequency": "DAILY",
  "body": {
    "FX.SPOT.EUR.USD.CITI": {
      "x": [20170320, 20170321, 20170322, 20170323, 20170324],
      "o": [1.0742, 1.0737, 1.0796, 1.0789, 1.0783],
      "h": [1.0778, 1.0812, 1.0825, 1.0806, 1.0818],
      "l": [1.0719, 1.0736, 1.0776, 1.0768, 1.0761],
      "c": [1.0737, 1.0796, 1.0789, 1.0783, 1.0799],
      "type": "SERIES"
    }
  },
  "status": "OK"
}
```

o & h & l

Corresponding Open, High, and Low values.

Note: unlike with the “x” & “c” arrays, the “o” & “h” & “l” arrays can contain nulls, though this is relatively rare.

Metadata API

<https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/metadata>

Provides metadata for the time series. The metadata is keyed by the same tags as the data. Currently we provide three types of metadata per tag: the tag description, the first and last dates available, and a history of series modification times. A modification *usually* means a new point has been appended to the series, so modification times are useful for estimating at what time of day new (EOD) data points become available.

Sample Request

```
{
  "tags": [
    "COMMODITIES.SPOT.SPOT_GOLD"
  ],
  "frequency": "EOD"
}
```

tags

An array of between 1 and 1000 distinct tags.

frequency

An optional parameter which can take one of two values: EOD or INTRADAY. EOD is the default.

Sample Response

```
{
  "body": {
    "COMMODITIES.SPOT.SPOT_GOLD": {
      "description": "Spot Gold Index Data",
      "modifiedTimes": [
        "2018-07-09T04:58:47-04:00[America/New_York]",
        "2018-07-06T04:48:50-04:00[America/New_York]",
        "2018-07-05T05:01:24-04:00[America/New_York]",
        "2018-07-04T04:52:32-04:00[America/New_York]",
        "2018-07-03T04:44:25-04:00[America/New_York]",
        "2018-07-02T05:11:21-04:00[America/New_York]",
        "2018-06-29T05:02:46-04:00[America/New_York]",
        ...
      ],
      "startDate": 20060102,
      "endDate": 20180706
    }
  },
  "status": "OK"
}
```

The `modifiedTimes` array will have between 1 and 10 values. The timestamp format is compatible with [ISO-8601](#). The `modifiedTimes` and `endDate` fields are only available for the EOD frequency.

Tag Listing API

`https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/taglisting`

Allows for listing out tags with a given prefix. For example, the prefix 'COMMODITIES.SPOT.' corresponds to these three tags:

```
COMMODITIES.SPOT.SPOT_GOLD
COMMODITIES.SPOT.SPOT_SILVER
COMMODITIES.SPOT.OIL_PRICE_NYMEX
```

Sample Request -1

```
{"prefix": "COMMODITIES.SPOT.", "regex": ".* (GOLD|SILVER)"}
```

prefix

A tag prefix. Must contain at least the first two fields ("Category" and "Sub Category" in the Data Browser). The trailing '.' is optional.

regex

An optional regular expression which further restricts the list of tags matching the specified prefix. The regex is matched against the entire tag. Begin the regex with '.*' to avoid repeating the prefix. The regex follows [Java conventions](#).

Q: Why do I need to provide the prefix separately from the regex?

A: Because there are millions of tags and testing each one against the regex is expensive.

Sample Response -1

```
{
  "tags": [
    "COMMODITIES.SPOT.SPOT_GOLD",
    "COMMODITIES.SPOT.SPOT_SILVER"
  ],
  "status": "OK"
}
```

Generic Tags

Certain tags are separated behind the scenes into a "generic" tag like 'CREDIT.BOND.<ISIN>.PRICE' and a security identifier like US459200JG74. This separation is needed when there are many securities with the same measure (e.g., thousands of bonds have a price). Generic tags will have placeholder fields surrounded by <>.

Sample request: `{"prefix": "CREDIT.BOND"}`

Get backtest only tag list

Sample Request -2

```
{ "prefix":"ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA", "tagType":"BT" }
```

tagType

parameter tagType with value “BT”, to get the backtest only tag list. These tags no longer updates and data is available for backtest only.

Sample Response -2

```
{
  "tags": [
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E110.FRA_E110_1.01495233639.ACTUAL",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E110.FRA_E110_1.0664065462.ACTUAL",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E28.FRA_E28_1.069822702.ACTUAL",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E28.FRA_E28_1.069822702.EXPECTED",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E28.FRA_E28_2.069822702.ACTUAL",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E28.FRA_E28_2.069822702.EXPECTED",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E28.FRA_E28_2.069822702.PREVIOUS",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E28.FRA_E28_3.069822702.ACTUAL",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E44.FRA_E44_1.069822702.PREVIOUS",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E81.FRA_E81_1.0664065462.ACTUAL",
    "ECONOMICS.DJ_ECO.DJ_EUR.DJ_FRA.FRA_E82.FRA_E82_1.0664065462.EXPECTED"
  ],
  "status": "OK"
}
```

Get identifier list for generic tag

Sample Request – 3

```
{ "prefix": "EQUITY.STOCK", "tagType": "Security" }
```

tagType

parameter tagType with value “Security”, to get the id list for given data set.

Support prefix

```
{EQUITY.STOCK}, {EQUITY.ETF}, {EQUITY.EQUITY_INDEX}, {EQUITY.EQIVOL.STOCK}, {EQUITY.EQIVOL.ETF}, {EQUITY.EQIVOL.EQUITY_INDEX}, {CREDIT.BOND}, {RATES.BOND}
```

Sample Response – 3

```
{
  "tags": [],
  "downloadlink":
  "https://www.citivelocity.com/marketdata/aknetdownload/securitydetails/EQUITY.STOCK.xlsx?hash=l8k9EpVad1lmEUV1SslIP-oDstThzqgo2ocYW1BRSUs=",
  "msg": "success",
  "status": "OK"
}
```

Tag Browsing API

<https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/tagbrowsing>

This API allows for exploring the tag tree structure/hierarchy one level at a time. It is geared towards enabling the creation of a simple Data Browser-like widget or UI.

Sample Requests

```
{ "prefix": "" }
{ "prefix": "COMMODITIES.SPOT" }
{ "prefix": "COMMODITIES.SPOT.SPOT_GOLD" }
```

prefix

This field can be thought of as selecting a node within the tag tree structure. Pass the empty string ("") for the root node.

Sample Responses

Root Node ("")

```
{
  "header": "Category",
  "fields": {
    "COMMODITIES": "Commodities",
    "CREDIT": "Credit",
    "ECONOMICS": "Economics",
    "EM": "Emerging Markets",
    "EQUITY": "Equities",
    "FX": "FX",
    "MUNI": "Municipals",
    "RATES": "Rates",
    "SECURITIZED": "Securitized"
  },
  "leaves": [],
  "status": "OK"
}
```

Interior Node ("COMMODITIES.SPOT")

```
{
  "header": "Product",
  "fields": {
    "OIL_PRICE_BRENT": "Brent Spot Month (ICE)",
    "OIL_PRICE_NYMEX": "WTI Spot Month (NYMEX)",
    "SPOT_GOLD": "Gold (Spot Price)",
    "SPOT_SILVER": "Silver (Spot Price)"
  },
  "leaves": [
    "OIL_PRICE_NYMEX",
    "SPOT_GOLD",
    "SPOT_SILVER",
    "OIL_PRICE_BRENT"
  ],
  "status": "OK"
}
```

Leaf Node ("COMMODITIES.SPOT.SPOT_GOLD")

```
{
  "fields": {},
  "description": "Spot Gold Index Data",
  "leaves": [],
  "status": "OK"
}
```

UI Implementation

The tree-structure / nodes should be cached on client side with some reasonable expiration time (e.g. 12 hours). Note that different users have access to different tags, so the cache should be per user.

Equity Identifier APIs

`https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/citiids/from`

`https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/authed/citiids/to`

These two APIs allow for converting between Citi Equity Identifiers and other “street” identifiers.

Refer to the new “User Defined Tags” functionality for a more convenient way to access Equity Products

Q: Why not use the standard symbols for equities?

A: Unfortunately, there is no widely accepted standard. Some shops use Bloomberg Tickers, others use Reuters Instrument Codes, etc. We also require that the identifier specifies *both* the instrument and the exchange. A Bloomberg Ticker or Exchange Traded Symbol like “IBM” fails on this account.

/from - Sample Request

```
{"ids": [123, 306888, 56280, 92141]}
```

ids

A list of up to 10000 integers (64-bit, long in Java). These are the Citi Equity Identifiers.

/from - Sample Response

```
{
  "body" : [ null, {
    "id" : 306888,
    "ric" : "IBM.N",
    "mic" : "XNYS",
    "ets" : "IBM",
    "bbt" : "IBM",
    "isin" : "US4592001014"
  }, {
    "id" : 56280,
    "ric" : "SN.L",
    "mic" : "XLON",
    "ets" : "SN.",
    "bbt" : "SN/",
    "isin" : "GB0009223206"
  }, {
    "id" : 92141,
    "ric" : ".SPX",
    "ets" : "SPX",
    "bbt" : "SPX",
    "isin" : "US78378X1072"
  } ],
  "status" : "OK"
}
```

body

The response contains an array of equal length to the `ids` array in the request. Each index in the response body corresponds to the same index in the request. The value at a given index in the response will be null if the requested Citi Id is not recognized (e.g. 123 in the example above), otherwise it will contain the following fields:

Field	Description
id	Citi Equity Identifier
ric	Reuters Instrument Code
mic	Market Identifier Code
ets	Exchange Traded Symbol
bbt	Bloomberg Ticker
isin	International Securities Identification Number

Note that the only field guaranteed to be present in the response is the (Citi) `id`. In particular, `mic` will often be missing for Indices (see SPX example above).

/to – Sample Request

```
{
  "queries": [
    {
      "productType": "STOCK",
      "identifier": "IBM.N",
      "identifierType": "RIC"
    },
    {
      "productType": "STOCK",
      "identifier": "IBM",
      "identifierType": "BBT",
      "mic": "XNYS"
    },
    {
      "productType": "STOCK",
      "identifier": "IBM",
      "identifierType": "BBT",
      "primaryOnly": true
    }
  ]
}
```

queries

A list of up to 1000 query objects. Each query should identify exactly one instrument and exchange combination. There are three ways to achieve this (corresponding to the three sample queries above):

1. Use an identifier with a built-in exchange. Currently RIC is the only such supported option.
2. Specify the exchange using the `mic` field. This approach only works well for Stocks and ETFs.
3. Specify that we only care about the primary exchange using the `primaryOnly` field.

Field	Optional	Description
productType	N	STOCK, ETF, or INDEX
identifier	N	
identifierType	N	RIC, BBT, ETS, or ISIN
mic	Y	see below for details
primaryOnly	Y	filter for primary exchanges when true

BBT warning

It has recently been brought to our attention that Bloomberg Tickers (BBT) are not unique across all regions/exchanges. For example, “PEP” is the ticker for PepsiCo Inc in the US, but Pepper Group Ltd in Australia.

RIC / NASDAQ warning

Please note the distinction between the “.O” and “.OQ” suffixes when working with RICs for stocks traded on one of the NASDAQ markets. Rule of thumb guide:

Mic	Market Description	RIC Suffix
XNMS	NASDAQ/NMS (GLOBAL MARKET)	.O
XNGS	NASDAQ/NGS (GLOBAL SELECT MARKET)	.OQ
XNCM	NASDAQ CAPITAL MARKET	.OQ

mic

Only works well for Stocks and ETFs. Must be one of the following supported exchanges (specifying a MIC which is not on this list will cause the entire request to fail):

ARCX, XAMS, XASE, XASX, XBRU, XCSE, XETR, XFRA
 XHEL, XHKG, XJPX, XLIS, XLON, XNAS, XNCM, XNGS
 XNMS, XNYS, XPAR, XSTO, XTKS

/to - Sample Response

```
{
  "ids" : [ 306888, 306888, 306888 ],
  "status" : "OK"
}
```

All three queries in the sample request correspond to the same Citi Equity Identifier.

ids

The response contains an array of equal length to the `queries` array in the request. Each index in the response corresponds to the same index in the request. The value at a given index in the response will be null if the corresponding query did not result in exactly one Citi Id. These are 64-bit integers (`long` in Java).

User Defined Tags

We are offering user defined tags for easier usage of equity and credit products.

Tags will be in format UDT.ADMIN.<Tag> where the Citi ID of equity/credit instruments replaced by RIC/BBT.

Support Data Set
EQUITY.STOCK
EQUITY.EQUITY_INDEX
EQUITY.EQIVOL.ETF
EQUITY.EQIVOL.STOCK
EQUITY.EQIVOL.EQUITY_INDEX
CREDIT.CDS

Sample Charting Tag → User Definer Tag
EQUITY.EQIVOL.EQUITY_INDEX.92141.EQIVOL_DELTA.STRIKE_C25.2W.CITI → UDT.ADMIN.EQUITY.EQIVOL.EQUITY_INDEX..SPX.EQIVOL_DELTA.STRIKE_C25.2W.CITI
EQUITY.EQUITY_INDEX.92141.LEVEL.REUTERS → UDT.ADMIN.EQUITY.EQUITY_INDEX..SPX.LEVEL.REUTERS
EQUITY.STOCK.306888.VOLUME.REUTERS → UDT.ADMIN.EQUITY.STOCK.IBM.N.VOLUME.REUTERS
CREDIT.CDS.MKS-M+SPLC.SNRFOR.EUR.MM14.5Y.PAR.BLENDED → UDT.ADMIN.CREDIT.CDS.MARSPE.SNRFOR.USD.XR14.5Y.PAR.BLENDED

(Special character “.” will be treated as underscore while processing)

Sample Request

Here is an example of how to pull data for user defined tags:

```
{
  "startDate": 20170323,
  "endDate": 20170323,
  "tags": [
    "UDT.ADMIN.EQUITY.STOCK.IBM_N.VOLUME.REUTERS"
  ],
  "startTime": 0,
  "endTime": 2359,
  "frequency": "HOURLY"
}
```

Metering

Various limits are in place to restrict the amount of data that can be pulled.

Request Size & History “Size”

As mentioned above, data requests can contain at most 100 tags per request, while metadata requests can contain up to 1000 tags per request. The amount of history is also limited for intraday frequencies (details above). Exceeding the former is treated as an error, while exceeding the latter results in the frequency being bumped up to a coarser one to reduce the size of the response.

Counting Limits

Type	Data API	Metadata	Tag Listing	Tag Browsing
Max Calls	10,000	10,000	100	1,000
Max Items	100,000	100,000	100,000	1,000
Max Calls Per Item	10	5	-	-

Type	Citi Id - From	Citi Id - To	Filter
Max Calls	10,000	10,000	1,000
Max Items	100,000	100,000	1,000,000
Max Calls Per Item	5	-	-

These limits reset roughly every 24 hours. Separate counts are maintained for each API. So you can make 10000 data calls and separately 10000 metadata calls. For Max Items we count the total number of tags (or ids) requested, not the number of distinct tags (or ids).

Bespoke tags created via the Vol & Swap Pricer are available via the Data API, but with stricter limits:

Max Items	1,000
Max Items Per Request	10
Max Calls Per Item	2

Rate Limits

Type	Limit	Description
Concurrency	1 thread	Only one request is processed at a time
Rate	1 per second	At most one request is processed per second

Both limits are per account – of course the service itself is multi-threaded and handles more than one request per second. Exceeding either limit causes subsequent requests to queue up until the limit is no longer exceeded. Queued requests time out after one minute.

Note that these are rate *limits*, not guarantees. The actual throughput is going to depend on many factors such as network traffic, application server utilization, and database utilization – some of which is beyond Citi’s control.

Q: Given the concurrency limit, is there any benefit to multi-threading the requests (against the same API)?

A: The short answer is “not really”. Two request threads might give a minor boost, but in practice this will not yield much improvement over a single-threaded client, and is probably not worth the additional effort and complexity. Using more than two threads will not yield any improvements, and may lead to requests timing out.

Q: Does my single-threaded client need to worry about the rate limits? Do I need to sleep in my client?

A: No. You can ignore the rate limits for simplicity. The server-side queuing is transparent to your client.

Testing

The following tags are excluded from the “Max Calls Per Item” limits, and can be used for testing:

`TEST.n.FX.FORWARD`, $n \in \{0..999\}$

Miscellaneous

New Fields in Response JSON

Existing fields will never be removed or renamed so as to maintain backwards compatibility. However, new fields can be added at any time without notice to the response JSON. Please make sure your JSON parsing is robust against the presence of new fields in the response.

HTTP Error Codes

Most errors are caught and handled at the JSON response level (i.e. status or type “ERROR”). So the HTTP response code will usually be 200. However, it is possible that some unexpected errors (or service downtime) will result in HTTP error codes ≥ 400 , and client code should handle that possibility.

Downtime and Retry Logic

This web service can be down for up to 10 minutes at any time during the day. Client code must have retry logic built in to handle short periods of downtime. Maintenance requiring more than 10 minutes of downtime will be scheduled for the 48 hour window between Friday 6pm and Sunday 6pm NY time.

gzip

Please verify that your client accepts the gzip Content-Encoding. This will greatly reduce network traffic.

Corporate Proxy

To check if you are behind a corporate proxy on Windows execute:

Start >> cmd >> ping www.citivelocity.com

“Request timed out.” indicates the presence of a proxy.

Sample Python

This sample code makes use of the “requests” HTTP library for Python. Installation instructions [here](#). Help with a corporate proxy [here](#).

```
>>> import requests
>>> url =
'https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/aut
hed/data?client_id=your_client_id'
>>> payload = {'startDate': 20170108, 'endDate': 20170114, 'tags':
['COMMODITIES.SPOT.SPOT_GOLD']}
>>> headers = {'authorization': 'Bearer your_access_token'}
>>> r = requests.post(url, json=payload, headers=headers)
>>> r.text
```

Sample Curl

The `curl` command line utility can be used to quickly test connectivity and the request JSON format. Below is a sample `curl` command corresponding to the first example above. See [here](#) for help with a corporate proxy.

```
curl -H 'Content-Type: application/json' -H 'authorization: Bearer
your_access_token' -X POST -d '{"startDate": 20170108, "endDate": 20170114,
"tags":["COMMODITIES.SPOT.SPOT_GOLD"]}'
'https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/aut
hed/data?client_id=your_client_id'
```

Sample R

```
library(httr)
library(jsonlite)

client_id <- "your_client_id"
access_token <- "your_access_token"
endpoint <-
'https://api.citivelocity.com/markets/analytics/chartingbe/rest/external/aut
hed/data'

url <- paste(endpoint, "?client_id=", client_id, sep="")

req <- '{"startDate": 20170108, "endDate":
20170114, "tags":["COMMODITIES.SPOT.SPOT_GOLD"]}'

resp <- POST(url, body=req, add_headers("Content-Type" = "application/json",
"authorization" = paste("Bearer ", access_token, sep="")))

resp_content <- content(resp, "text")

resp_content
```