MORNINGSTAR®

MarketPlace API

User Guide

Publish date: January 2019

Table of Contents

Overview

The Marketplace site hosts both a web interface and a REST based API. This document covers the REST based API. Before diving into details about the API, some terminology around the marketplace is explained.

Feeds

The Marketplace data is organized into feeds. A "feed" is a set of data that is bundled and licensed together by a data vendor. It is identified by a unique string, which we call the feed name. Much like a table in a relational database, a Marketplace feed contains a set of columns where one or more of the columns form a primary key and the remaining columns are data-value columns. Because the feeds hold timeseries information, all feeds implicitly have an additional DateTime field. As an example, a weather feed may have key columns "City" and "State" and data columns "HighTemp" and "LowTemp." Likewise, an equities feed may have key column "Ticker" and data columns "Open", "High", "Low", and "Close." In general, the column names and keys used in a feed matches as closely as possible with the original vendor source data.

User Accounts

User accounts in the Marketplace are identified by the users' email address. When a user is licensed for a data feed, the user is subscribed to that feed. The user will only be able to access subscribed feeds. All API access requires HTTP BASIC authentication with the user's login credentials. To ensure credentials are transmitted securely, we recommend using HTTPS with the API.

Extracting Data

Currently, the APIs for extracting data from a feed are pull-based. The pull request can include historical data, point-in-time views of the data, and corrections history. The general form for API calls includes the URL to the resource being requested, HTTP Basic user credentials, and an HTTP Accept header that indicates the desired output format. All API calls support MIME type "application/json" output. There are some APIs that provide other MIME types for convenience. Any optional arguments to an API are supported thru HTTP query parameters. Any API calls that accept dates as input, or that produce dates as output, will be in ISO 8601 format unless otherwise specified. In ISO 8601, dates are specified as yyyy-mm-dd. Date-times are specified as yyyy-mm- ddThh:mm:ss where a capital letter 'T' separates the date from the time field.

File Browsing and Retrieval

To view a list of files that are available for download from a feed, you can use the Feed Items resource.

Feed Items Resource	
URL	https:// mp.morningstarcommodity.com/lds/feeds/{feedname}/items
Description	Fetches a list of files for a feed.
Method	HTTP GET
Request Payload	None
Response Payload	JSON String
Accept Header	Application/json

There are no required query parameters to the above URL, but there are some optional query parameters.

Feed Items Resource	
type	Allows the user to filter files based on file type. The allowable types are all, src, cf, and delta. The default is all.
time	Allow the user to filter out any files that arrived before
	the specified time. The format of the time parameter is an ISO 8601 string. If not specified, this filter is not applied.
endtime	Allow the user to filter out any files that arrived after the specified time. The format of the time parameter is an ISO 8601 string. If not specified, this filter is not applied.
limit	To limit the response of the number of entries. Default is -1 (which pulls back everything)
skip	To skip the number of values. The default value is 0

Sample input:

```
GET /lds/feeds/ECB_EuroFxRef/items?time=2011-10-17 HTTP/1.1
Authorization: Basic abctZHdAGbltLmnvbTqspC3ZcWlt
Host: mp.morningstarcommodity.com
Accept: application/json
```

Sample output:

```
HTTP/1.1 200 OK
Content-Type: application/json
Date: Tue, 18 Oct 2011 14:21:59 GMT
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=644DFDFE53EFC967614BF465F6F581D9; Path=/lds
Content-Length: 1841
Connection: keep-alive
[
{
"feedName":"ECB_EuroFxRef",
"itemId":"src_LIM_ECB_EuroFxRef_csv_20111017_074413.zip",
"length":521,
"md5checksum":"a235f8183dbc187aac1c987cedc1787d",
"storeTime":"2011-10-17T07:45:35-05:00",
"type":"src",
"uri":"http://mp.morningstarcommodity.com/lds/feeds/ECB_EuroFxRef/items/
src_LIM_ECB_EuroFxRef_csv_20111017_074413.zip"
}
]
```

For each item listed in the output, the itemId indicates the filename. The url parameter indicates a link that can be used to fetch the file. The length and md5checksum contain further useful information about the file.

Fetch Individual File

To fetch an individual file from the Marketplace, you pull directly from the feed item resource.

Feed Items Resource	
URL	https://mp.morningstarcommodity.com/lds/feeds/{feedname}/items/filename.zip
Description	Fetches an individual file
Method	HTTP GET
Request Payload	None
Response Payload	File bytes
Accept Header	Application/octet-stream

Sample input:

```
GET /lds/feeds/ECB_EuroFxRef/items/src_file_csv_20111017_074413.zip HTTP/1.1
Authorization: Basic cBltZHdAbXXabcNvbTpsaW1YbXyz
Host: mp.morningstarcommodity.com
Accept: application/octet-stream
```

Sample output:

```
Content-Disposition: attachment; filename="src_file_csv_20111017_074413.zip"
Content-Type: application/octet-stream
Date: Tue, 18 Oct 2011 14:40:32 GMT
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=A1799DF856A9CC08FE1EAA3C5FC738E8; Path=/lds
Content-Length: 521
Connection: keep-alive
[ raw binary data not shown ]
```

Extracting Keys

The keys resource allows a user to examine and search for the primary keys for a particular feed.

Key Resource	
URL	https://mp.morningstarcommodity.com/lds/feeds/{feedName}/keys
Description	Fetches all the possible keys that match the input criteria. If no query parameters are provided, it will fetch all keys for the specified feed.
Method	HTTP GET
Request Payload	None
Response Payload	A JSON document that contains key/value pairs indicating the keyname and keyvalue.
Accept Header	Application/json

Optional Query Parameters for Key Resource	
limit	Allows the user to limit the number of results returned. For example, limit=20 would limit the result list to the first 20 matching hits. The default value for limit is 10. If set to -1, the results are not limited.
page	To page the results.
{keyName}	Allows the user to filter the results returned. For example, if you have a feed with key "ticker", and you were interested in keys that have ticker starting with the letter M. In this case, the query parameter would be "ticker=M".

Sample input:

```
GET /lds/feeds/ECB_EuroFxRef/keys?limit=-1 HTTP/1.1
Authorization: Basic bXltAHdAbzztLmNvbTpsaW1YbWls
Accept: application/json
```

Sample output

```
HTTP/1.1 200 OK
Cache-control: no-cache="set-
cookie" Content-Type:
application/json
Date: Fri, 03 Feb 2012 03:26:47 GMT
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=6398F59820C65370875A240D3CA4EA1A; Path=/lds
Content-Length: 1734
Connection: keep-
alive
{"matche":[{"keys":{"key":"Currency","value":"AUD"}},{"keys":
{"key":"Currency","value":"BGN"}},{"keys":{"key":"Currency","value":"BRL"}},{"k
eys":
{"key":"Currency","value":"CAD"}},{"keys":{"key":"Currency","value":"CHF"}},{"k
eys":
{"key":"Currency","value":"CNY"}},{"keys":{"key":"Currency","value":"CYP"}},{"k
eys":
{"key":"Currency","value":"CZK"}},{"keys":{"key":"Currency","value":"DKK"}},{"k
eys":
{"key":"Currency","value":"EEK"}},{"keys":{"key":"Currency","value":"GBP"}},{"k
eys":
{"key":"Currency","value":"HKD"}},{"keys":{"key":"Currency","value":"HRK"}},{"k
eys":
{"key":"Currency","value":"HUF"}},{"keys":{"key":"Currency","value":"IDR"}},{"k
eys":
{"key":"Currency","value":"ILS"}},{"keys":{"key":"Currency","value":"INR"}},{"k
eys":
{"key":"Currency","value":"ISK"}},{"keys":{"key":"Currency","value":"JPY"}},{"k
eys":
{"key":"Currency","value":"KRW"}},{"keys":{"key":"Currency","value":"LTL"}},{"k
eys":
{"key":"Currency","value":"LVL"}},{"keys":{"key":"Currency","value":"MTL"}},{"k
eys":
{"key":"Currency","value":"MXN"}},{"keys":{"key":"Currency","value":"MYR"}},{"k
eys":
{"key":"Currency","value":"NOK"}},{"keys":{"key":"Currency","value":"NZD"}},{"k
eys":
{"key":"Currency","value":"PHP"}},{"keys":{"key":"Currency","value":"PLN"}},{"k
eys":
{"key":"Currency","value":"ROL"}},{"keys":{"key":"Currency","value":"RON"}},{"k
eys":
{"key":"Currency","value":"RUB"}},{"keys":{"key":"Currency","value":"SEK"}},{"k
eys":
{"key":"Currency","value":"SGD"}},{"keys":{"key":"Currency","value":"SIT"}},{"k
eys":
{"key":"Currency","value":"SKK"}},{"keys":{"key":"Currency","value":"THB"}},{"k
eys":
{"key":"Currency","value":"TRL"}},{"keys":{"key":"Currency","value":"TRY"}},{"k
eys":
{"key":"Currency","value":"USD"}},{"keys":{"key":"Currency","value":"ZAR"}}]}
```

Pulling Forward Curves

List Resource	
URL	https://mp.morningstarcommodity.com/lds/feeds/{feedName}/curve?root={Root}&cols={Comma-Separated-Columns}&date={YYYY-MM-DD}
Description	Fetches a futures curve for specified root and date
Method	HTTP GET
Request Payload	None
Response Payload	JSON file
Accept Header	application/json

There are no optional query parameters for the list resource

The formula JS API can also be pulled back using a curl. Example:

Create a file with this content:

```
var feed ="LME_ClosingPrice";
var StepType = "Month";
var a=[];
var root0=["CAD"];
var column0="Last_Price";
var dateOfCurve0 = "2018-05-29";
var c0=forward_curve(feed,root0,column0,dateOfCurve0,StepType);
c0.toCsv(true)
```

Save the file and name it what you want (data.js)

run the curl command with the js formula

```
curl -n "https://mp.morningstarcommodity.com/lds/formulas/run/js" --data-binary @data.js
-H "Content-Type:text/plain" -H "Accept:text/csv"
```

Publisher Lists

If you have configured a Publisher list via the marketplace web interface, you can use REST calls to view the Publisher list configuration or to pull the data associated with the Publisher list.

List Details

To examine a Publisher list's details, use the list resource. The list name is the name of the Publisher list that was initially created.

List Resource	
URL	https://mp.morningstarcommodity.com/lds/lists/{listName}
Description	Fetches the detail information about the list
Method	HTTP GET
Request Payload	None
Response Payload	A string containing JSON that describes the list
Accept Header	Application/json

There are no optional query parameters for the list resource.

List Configuration

To examine a Publisher list's configuration, use the list resource. The list name is the name of the Publisher list that was initially created.

List Resource	
URL	https://mp.morningstarcommodity.com/lds/lists/{listName}?withMappings=true&withPubCols=true&wrap=false
Description	Fetches the configuration information about the list, including mappings
Method	HTTP GET
Request Payload	None
Response Payload	A string containing JSON that describes the list
Accept Header	Application/json

There are no optional query parameters for the list resource.

Pull List Content

The "list content" resource allows you to pull the data associated with the Publisher list. If no content exists, place a "?" after the word 'content'.

Example: http://mp.morningstarcommodity.com/lds/lists/{listName}/content?

List Content Resource	
URL	https://mp.morningstarcommodity.com/lds/lists/{listName}/content
Description	Fetches any data-points that have changed for each item in the Publisher list
Method	HTTP GET
Request Payload	None
Response Payload	A CSV/XML that contains the data point that has changed
Accept Header	None required. Defaults to CSV output.

When pulling the list, you can specify a time range of points so that you can see any new points or corrections that have arrived during that window of time.

Optional Query Parameters for Key Resource	
fromDateTime	Only fetch data-points that have changed since this time. The time format should be specified as an ISO 8601 date-time string in UTC.
toDateTime	Only fetch data-points that have changed before this time. The time format should be specified as an ISO 8601 date-time string in UTC.
delim	Allows the user to control the delimiter used in the output. The default delimiter is the ',' comma character.
daysBack	Set the number of days back the history should go to. Setting the parameter to "1" will go back 1 day from today.
backDateTime	Only displays records that occurred after the specified time.
corrections=update	This will only display corrections for values in your publisher list. Corrections=insert will limit the results to inserts only.
fromPubDateTime	Only fetches data for transaction dates after this time. The time format should be specified as an ISO 8601 date-time string in UTC.

toPubDateTime	Only fetches data for transaction dates before this time. The time format should be specified as an ISO 8601 date-time string in UTC.

Example: http://mp.morningstarcommodity.com/lds/lists/ECB/content?fromDateTime=2012-10- 20&toDateTime=2012-10-25

The publisher output is in CSV format. There are a number of fields that are extracted by default. The following table describes each field.

Field	Description
Type	Type of delta file - delete, insert, update
TSID	The time series ID for the given key values for the line in the output. This is for tracking in the time series database only.
Keys	The value for each key (e.g. NG5F)
KEY_NAMES	The name for each key (e.g. PNodeID)
COL_NAMES	All the available column values for the feed
NVInsertionDT	All new data insertion date and time
NewValues	All new data values
PVInsertionDT	Previous data insertion date and time before a correction
PreviousValues	Previous data values before a correction
PUBREASON	Reason the point was published
PUBCOL	The name of the column for the PUBVAL ie: Settlement Price
PUBVAL	The publication value
PUBDATE	The publication date
PUBMD	The publication metadata requested
PUBCMD	The publication company metadata requested
PUBLIST	Name of the list that contained the key or regular Expression
PUBFEED	The feed name for a key
MATCHPATTERN	If the PUBREASON was because of a pattern/regular expression, this will contain that expression
PUBSOURCE	The name of the original source of the data
PUBDATE_TZ	Published date time zone

Create a Publisher List

Create a publisher list by specifying the output parameters.
POST https://mp.morningstarcommodity.com/lds/lists

{
"delimiter": ",", "description": "my new list", "dtformat": "dd/MM/yyyy", "header": "false",

"name": "newlist1", "numberformat": "###.##", "pubFormat": "csvnoheader",
}

Parameters	
delimiter	Delimiter for the output. Choose from comma (,), pipe(|), colon(:), space, or tab
description	Description of the list
dtformat	Date format ex: yyyy-mm-dd
header	Header included in the output, choose from true or false
name	Name of the list
numberformat	Format of the number. Example: ###.## or ###,##
pubFormat	Format of the publisher list. Choose from csv, csvnoheader, or xml

Sample input

```
POST /lds/lists HTTP/1.1
Host: mp.morningstarcommodity.com
Content-Type: application/json
Cache-Control: no-cache

{
"delimiter": ",",
"description": "my new list",
"dtformat": "yyyy-MM-dd",
"header": "true",
"name": "newlist1",
"numberformat": "###.##",
"pubFormat": "csv",
}
```

Sample output:

```
STATUS 200 OK
```

Add Data to a Publisher List

Once a publisher list has been created, a user can add keys to the publisher list.

POST https://mp.morningstarcommodity.com/lds/lists/newlist1/rows

[
{
"columns": "column_name",
"feedName": "feed_name",
"keysMap": {"keys1":"keys2"},
"type": "TS"
}
]

Parameters	
columns	Name of the column that needs to be added. Make sure the column exists before adding to the list
feedName	Name of the feed
keysMap	Keys that need to be added
type	TS for time series; WILD for all keys using regular expressions

Sample input

```
POST /lds/lists/newlist1/rows HTTP/1.1
Host: mp.morningstarcommodity.com
Content-Type: application/json
Cache-Control: no-cache

[
{
"columns":  "Rate",
"feedName": "ECB_EuroFxRef",
"keysMap": {"Currency":"AUD"},
"type": "TS"
},
{
"columns":  "Rate",
"feedName": "ECB_EuroFxRef",
"keysMap": {"Currency":"BGN"},
"type": "TS"
},
{
"columns": "Rate",
"feedName": "ECB_EuroFxRef",
"pattern": "C.*",
"type": "WILD"
}
]
```

Sample output:

```
STATUS 200 OK
```

Only include specific rows

Users can include specific rows from the results by retrieving the row "identity". First the user must retrieve the list.

```
"http://mp.morningstarcommodity.com/lds/lists/{listname}"

{      "rows": [
       {
       "columns": "Rate",
       "companyMdColumns": "stevecol3",
       "feedName": "ECB_EuroFxRef",
       "identity": "df80d7316e46a442f7b4d4b424faca15f9d0f0ef",
       "keys": "AUD",
       "tsid": "ad4cecff-9a5b-4c64-a002-305ab8e8be50",
       "type": "TS"
       },
       {
       "columns": "Rate",
       "companyMdColumns": "stevecol3",
       "feedName": "ECB_EuroFxRef",
       "identity": "907e00a79c82c910131c7498c0f57d626514ddc1",
       "keys": "BGN",
       "tsid": "3fd8938a-e1ab-40be-a584-afccdb0c4a0b", "type": "TS"
} … etc …
```

Next, build a json object which includes the identity of the rows.

{ "includeRows":[“identity”, “identity”, …etc...] }

Example

```
{ "includeRows":[“df80d7316e46a442f7b4d4b424faca15f9d0f0ef”,
“907e00a79c82c910131c7498c0f57d626514ddc1”] }
```

There is a new POST method for the lists/{listname}/content that will take the json object in the payload of the request.

Here is a sample curl command (includeRows.json is a file that contains the above json)

```
-XPOST -H"Accept: text/csv"-H"Content-Type: application/json" -
d@includeRows.json
"http://mp.morningstarcommodity.com/lds/lists/ecbtest/content?fromDateTime=2015-
08-10 ”
```

Publisher Worker API

This resource allows you to specify the amount of history you want to run for a Publisher list. It can also be used to extract the “Last Value” if multiple values have been inserted for a data point within a specified time period. API supports all parameters that the “List Contents” API references on Page 11. Multiple requests cannot be processed for the same list and current processing ticket ID is returned when a new request is submitted while a list is still processing.

Submit Ticket

Resource URL

POST: https://mp.morningstarcommodity.com/lds/lists/{ListName}/tickets?fromDateTime=YYYY-MM-DD&toDateTime=YYYY-MM-DD

Note: Add additional parameters by separating them with “&”

Optional Query Parameters for List Content Resource	
fromDateTime	Only fetch data from delta files dated after this time. The time format should be specified as an ISO 8601 date-time string in UTC.
toDateTime	Only fetch data from delta files dated before this date. The time format should be specified as an ISO 8601 date-time string in UTC.
delim	Allows the user to control the delimiter used in the output. The default delimiter is the ‘,’ comma character.

daysBack	It is possible that corrections (or history) were loaded into one day's delta file. Setting this parameter would limit the amount of history (or corrections) to X number of days before the FromDateTime you are pulling.
backDateTime	Only displays records that occurred after the specified time
corrections=update	This will only display corrections for values in your publisher list. Corrections=insert will limit the results to inserts only.
fromPubDateTime	Only fetch data for transaction dates after this time. The time format should be specified as an ISO 8601 date-time string in UTC.
toPubDateTime	Only fetch data for transaction dates before this time. The time format should be specified as an ISO 8601 date-time string in UTC.

Response

Response Format: JSON
Response Parameters:
ticketId : UUID that uniquely identifies the requests that are submitted for processing
status: Indicates the status of the request. PENDING, DONE, ERROR
creationTime: Time indication the creation time of the request

Example

API Request:
Content-Type: application/json
POST : https://mp.morningstarcommodity.com/lds/lists/{ListName}/tickets?fromDateTime=2017-09-01&toDateTime=2017-09-09

Example Response:


```
{
   "creationTime": "2017-09-07T08:56:04.990-05:00",
   "status": "PENDING",
   "ticketId": "38e1f907-c956-42b1-bd10-83e9697f50d1"
}
```


Ticket Status

The "Ticket Status" resource will show you if a ticket/job is "Pending" or "Done" (status= "Pending" indicates the worker is still generating the requested file)

Resource URL

GET : https://mp.morningstarcommodity.com/lds/lists/tickets/{ticketId}
ticketID: UUID obtained from the List ticket API response

Response

Response Format: JSON
Response Parameters:
ticketId : UUID that uniquely identifies the requests that are submitted for processing

status: Indicates the status of the request. PENDING, DONE, ERROR
creationTime: Indicates the creation time of the request

Example
GET : https://mp.morningstarcommodity.com/lds/lists/tickets/38e1f907-c956-42b1-bd10-83e9697f50d1

Example Response:

```
{
   "creationTime": "2017-09-07T08:56:05-05:00",
   "status": "DONE",
   "ticketId": "38e1f907-c956-42b1-bd10-83e9697f50d1"
}
```


List Content

The "List Content" resource returns the publisher list output if the status of the ticket is "Done".

Resource URL
GET: http://mp.morningstarcommodity.com/lds/lists/{ListName}/content?ticketId=38e1f907-c956-42b1-bd10-83e9697f50d1

ListName: Publisher list name for which the output needs to be extracted
ticketId: ticketId obtained from the list ticket resource

Response
Publisher output response

Example
API Request:
Content-Type: Depending on the publisher output section in marketplace
GET: http://mp.morningstarcommodity.com/lds/lists/{listName}/content?ticketId=38e1f907-c956-42b1-bd10-83e9697f50d1

List Tickets

To see a list of all tickets that have been submitted for a list (with descending order of the creation time), use the below resource

Resource URL
GET https://mp.morningstarcommodity.com/lds/lists/{ListName}/tickets

ListName: publisher list name for which tickets needs to be retrieved

Response
Response Format: JSON

Response Parameters:
List of ticket status response with creation time, status, ticketId

Pull Timeseries Data

To extract a timeseries from a feed, use the "ts" resource.

Ts Resource	
URL	https://mp.morningstarcommodity.com/lds/feeds/{feedname}/ts? key1=value1 & key2=value2…
Description	Fetches a timeseries from the database
Method	HTTP GET
Request Payload	None
Response Payload	A string containing comma separated values
Accept Header	Text/csv
insertTimes	If true, it will give the insertion times for all the data points in the time series
dateFormat	The format of the date you want to pull back. Example: YYYY-MM-DD

In the above URL, notice that there are some required query parameters. The actual query parameters to use will depend on the keys that are defined for that particular feed. If we were pulling from the example weather feed described above, the query parameters would be ?City="Chicago"&state="IL", for example. If we were pulling from the example equity feed, the query parameters would be ?Ticker="MORN", for example.

Optional Query Parameters for Ts Resource	
fromDateTime	Filters the output by removing any rows in the output before the fromDateTime. This date format should be in ISO 8601.
toDateTime	Filters the output by removing any rows in the output after the toDateTime. This date format should be in ISO 8601.
asOfDateTime	This parameter allows the user to view a timeseries as it would have appeared at a particular point in time. If not specified, the most current timeseries is fetched.
delimiter	Allows the user to specify an alternate delimiter in the output. If not specified, a comma is used as the delimiter.

insertTimes	Setting insertTimes=true modifies the output to include an insertTime column for each series that is extracted. The default value for this parameter is false.
dateFormat	Allows the user to specify an alternate date format in the output. If not specified, the date format will be displayed in ISO 8601. The pattern for this field must conform to the patterns specified in Java 6 SimpleDateFormat.
cols	Allows the user to restrict the number of columns that are returned. If not specified, all the value columns for the feed are returned.
descending	When set to "true" the values will be sorted in descending order by date
maxResults	Sets the number of maximum results to be returned. Setting this number to 1 will return the most recent date and value.

Daily Sample input:

```
GET /lds/feeds/ECB_EuroFxRef/ts?Currency=USD&fromDateTime=2011-10-10 HTTP/1.1
Authorization: Basic cXltMHdAbXXlLmabcbTpsaW1YbWxt
Host: mp.morningstarcommodity.com
Accept: text/csv
```

Daily Sample output:

```
HTTP/1.1 200 OK
Content-Type: text/csv
Date: Tue, 18 Oct 2011 15:08:41 GMT
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=4EA7B9F5CE66AEA91865049E99E5E880; Path=/lds
Content-Length: 136
Connection: keep-alive

Date,rate
2011-10-10,1.3593
2011-10-11,1.3607
2011-10-12,1.3766
2011-10-13,1.3727
2011-10-14,1.3807
2011-10-17,1.3776
2011-10-18,1.3676
```

Hourly Sample input:

```
GET
/lds/feeds/ERCOT_SettlementPointPrices/ts?cols=SettlementPointPrice&timezone=U
S/Central&SettlementPointName=HB_NORTH&SettlementPointType=HU&toDateTime=2015-
07-19&fromDateTime=2015-07-19 HTTP/1.1
Host: mp.morningstarcommodity.com
Accept: text/csv
Cache-Control: no-cache
```

Hourly Sample output:

```
Date,SettlementPointPrice(HB_NORTH;HU)
2015-07-19T00:00-0500,22.52
2015-07-19T00:15-0500,24.81
2015-07-19T00:30-0500,21.05
2015-07-19T00:45-0500,21.15
2015-07-19T01:00-0500,20.79
2015-07-19T01:15-0500,20.37
2015-07-19T01:30-0500,19.73
2015-07-19T01:45-0500,19.3
2015-07-19T02:00-0500,19.16
```

Ordering data

Data can be displayed in ascending order by using the "descending" parameter.

Sample input:

```
GET /lds/feeds/ECB_EuroFxRef/ts?Currency=JPY&descending=true HTTP/1.1
Host: mp.morningstarcommodity.com
Cache-Control: no-cache
```

Sample output:

```
{
"series": [
{
"dates": [
"2015-07-07",
"2015-07-06",
"2015-07-03",
"2015-07-02",
"2015-07-01",
"2015-06-30",
"2015-06-29",
```

Limiting data

The number of results can be limited by using the "MaxResults" parameter. Below the max result is set to 1 which will only return the last data day.

Sample input:

```
GET /lds/feeds/ECB_EuroFxRef/ts?Currency=JPY&descending=true&maxResults=1
HTTP/1.1
Host: mp.morningstarcommodity.com
Cache-Control: no-cache
```

Sample output:


```
{
"series": [
    {
       "dates": [
          "2015-07-07"
      ],
      "values": [
          {
              "item": "133.88"
          }
      ]
    }
  ]
}
```

Upload

Users can upload proprietary data to the MarketPlace using the MarketPlace API.

There are two options to upload data:

1) partial=true
2) no partial option

The partial=true option enables a user to provide partial corrections for single values within a multiple value feed. If a feed has multiple columns, then partial=true does not require all columns to have values. Some columns that are not applicable can remain empty. If this option is not enabled, then all columns will require an input value.

Partial Option

1. Create feed with columns (hightemp and lowtemp)

```
curl -H"Accept: text/csv" -u user@company.com:password
"http://localhost:8080/lds/feeds/user_weather/ts?City=Austin&State=TX"
Date,hightemp(Austin;TX),lowtemp(Austin;TX)
2006-12-31T18:00-0600,100.0,71.1
2006-12-31T19:00-0600,100.0,71.2
2006-12-31T20:00-0600,100.0,71.3
2006-12-31T21:00-0600,100.0,71.4
2006-12-31T22:00-0600,100.0,71.5
2006-12-31T23:00-0600,100.0,71.8
2007-01-01T00:00-0600,100.0,71.9
2007-01-01T01:00-0600,100.0,71.1
2007-01-01T02:00-0600,100.0,71.11
2007-01-01T03:00-0600,100.0,71.12
<snip>
2013-07-27T02:00-0500,100.0,72.1
2013-07-27T03:00-0500,100.0,72.11
2013-07-27T04:00-0500,100.0,72.12
2013-07-27T05:00-0500,100.0,72.15
2013-07-27T06:00-0500,100.0,72.16
2013-07-27T07:00-0500,100.0,72.17
2013-07-27T08:00-0500,100.0,72.18
2013-07-27T09:00-0500,100.0,72.19
2013-07-27T10:00-0500,100.0,72.22
2013-07-27T11:00-0500,100.0,72.23
2013-07-27T12:00-0500,100.0,72.24
2013-07-27T13:00-0500,100.0,72.25
2013-07-27T14:00-0500,100.0,72.26
2013-07-27T15:00-0500,100.0,72.29
2013-07-27T16:00-0500,100.0,72.3
2013-07-27T17:00-0500,100.0,72.31
2013-07-27T18:00-0500,100.0,72.31
```

2. Create a new cdf file (file name: change1.cdf)

```
cat change1.cdf
Def:city*, state*, Date, Hightemp, Lowtemp
Austin,TX,2013-07-27T17:00-0500,,71.543
```

(Hightemp is left blank. with partial=true, the current value of 100.0 will remain unchanged)

3. Zip the file

```
zip change2.zip change2.cdf
  adding: change2.cdf (deflated 4%)
```

4. Post change with newoption

```
curl -XPOST --data-binary @change2.zip -H"Content-type: application/octet-stream" –u user@company.com:password
"http://localhost:8080/lds/providers/LIM/items?id=change2.zip&feed=c raig_weather&type=cf&partial=true"
```

5. Re-Pull the list

```
curl -H"Accept: text/csv" -u user@company.com:password
"http://localhost:8080/lds/feeds/user_weather/ts?City=Austin&State=TX"
Date,hightemp(Austin;TX),lowtemp(Austin;TX) 2006-12-31T18:00-0600,100.0,71.1
2006-12-31T19:00-0600,100.0,71.2
<snip>
2013-07-27T11:00-0500,100.0,72.23
2013-07-27T12:00-0500,100.0,72.24
2013-07-27T13:00-0500,100.0,72.25
2013-07-27T14:00-0500,100.0,72.26
2013-07-27T15:00-0500,100.0,72.29
2013-07-27T16:00-0500,100.0,72.3
2013-07-27T17:00-0500,100.0,71.543 <<< only changed one value
2013-07-27T18:00-0500,100.0,72.31
```

No Partial Option

Using the same values that were created in step 1, a change will be created using the no partial option.

1. Create a new cdf (change2.cdf)

```
cat change2.cdf
Def:city*, state*, Date, Hightemp, Lowtemp
Austin,TX,2013-07-26T17:00-0500,,99.999
```

(Hightemp is left blank. with partial=true not activated, the current value of 100.0 will change to NaN)

2. Zip the file

```
zip change2.zip change2.cdf
adding: change2.cdf (deflated 4%)
```

3. Post change without option

   curl -XPOST --data-binary @change2.zip -H"Content-type: application/octet-stream" -u user@company.com:password "http://localhost:8080/lds/providers/LIM/items?id=change2.zip&feed=c raig_weather&type=cf"

4. Re-pull the list

   curl -H"Accept: text/csv" -u user@company.com:password "http://localhost:8080/lds/feeds/user_weather/ts?City=Austin&State=TX"
   Date,hightemp(Austin;TX),lowtemp(Austin;TX)
   2006-12-31T18:00-0600,100.0,71.1
   2006-12-31T19:00-0600,100.0,71.2
   <snip>
   2013-07-27T11:00-0500,100.0,72.23
   2013-07-27T12:00-0500,100.0,72.24
   2013-07-27T13:00-0500,100.0,72.25
   2013-07-27T14:00-0500,100.0,72.26
   2013-07-27T15:00-0500,100.0,72.29
   2013-07-27T16:00-0500,100.0,72.3
   2013-07-27T17:00-0500,NaN,99,999 <<<<SUCCESS Changed ALL values

Appendix

Accessing Marketplace API with C#

Below is a sample code for accessing the Marketplace API using C# language. This is a generic example to show how a connection and retrieval of data can be executed.

```
//e.g.
https://mp.morningstarcommodity.com/lds/feeds/ECB_EuroFxRef/ts?Currency=AUD,CAD&fromDateTime=2014-05-04&toDateTime=2014-05-09T23:59:59&cols=Rate&asOfDateTime=2014-05-08&insertTimes=true&timezone=UTC&desc=true
        public static void GetTimeSeriesAsync()
        {
                var mpURLBase = "https://mp.morningstarcommodity.com/lds";
                var feedNameURL = "/feeds/ECB_EuroFxRef";
                var KeysQuery = "Currency=AUD,CAD";
                var FromDateStr = "fromDateTime=2014-05-04";
                var ToDateStr = "toDateTime=2014-05-09T23:59:59";
                var FeedColumnStr = "cols=Rate"; //comma delimited if multiple columns
                var AsOfDateStr = "asOfDateTime=2014-05-08"; // asOfDateTime=2014-05-08 or blank
                var InsertTimeStr4URL = "insertTimes=true"; //if you want to include InsertTimes in result otherwise blank
                var SelectedTimeZoneStr = "timezone=UTC"; //timezone=UTC, can be omitted. Only for non-daily data
                var IncludeDesc = "desc=true";
                string m_strUrl = String.Concat(
                        mpURLBase, //https://mp.morningstarcommodity.com/lds
                        feedNameURL, // ECB_EuroFxRef
                        KeysQuery, // Currency=AUD,CAD
                        FromDateStr, //fromDateTime=2014-05-04
                        ToDateStr, //toDateTime=2014-05-09T23:59:59 or toDateTime=2014-05-09
                        FeedColumnStr, // cols=Rate
                        AsOfDateStr, // asOfDateTime=2014-05-08 or omit
                        InsertTimeStr4URL, //insertTimes=true, if you want to include insertTimes in result
                        SelectedTimeZoneStr, //timezone=UTC, can be omitted. Only for non-daily data
                        IncludeDesc); //desc=true
                var username = "your username to LDS" ; //your credentials to Marketplace
                var password = "your password";
                var httpRequest = GetHttpWebRequestForLDS(m_strUrl, username, password, "text/csv");
                httpRequest.Timeout = 1800000;
                httpRequest.BeginGetResponse(GetResponseCallback, new object[] { httpRequest, eventHandler });
        }
public static void GetResponseCallback(IAsyncResult asyncResult)
        {
                var obj = (object[])asyncResult.AsyncState;
                var request = (HttpWebRequest)obj[0];
```

```
                try
                {
                        var response = request.EndGetResponse(asyncResult);
                        Stream responseStream = response.GetResponseStream();
                        if (responseStream != null)
                {
                        var streamReader = new StreamReader(responseStream);
                        var rawData = streamReader.ReadToEnd();
//raw data is csv, you need parse it.
                //The first line is header
/*e.g. Date,InsertTime,Rate(AUD | ECB European Central Bank, Currency: AUD - Australian Dollar per
Euro),InsertTime,Rate(CAD | ECB European Central Bank, Currency: CAD - Canadian Dollar per Euro)
2014-05-05,2014-05-05T07:34:16.811-0500,1.4975,2014-05-05T07:34:16.811-0500,1.5242
2014-05-06,2014-05-06T07:33:14.678-0500,1.4932,2014-05-06T07:33:14.678-0500,1.5232
2014-05-07,2014-05-07T07:30:17.578-0500,1.4909,2014-05-07T07:30:17.578-0500,1.5164
*/
                }
        }
        catch (Exception ex)
        {
        }
}
public static HttpWebRequest GetHttpWebRequestForLDS(string strUrl, string username, string password, string
acceptHeader)
        {
                var credentialCache = new CredentialCache();
                credentialCache.Add(new Uri(strUrl), "Basic", new NetworkCredential(username, password));
                var httpRequest = (HttpWebRequest)WebRequest.Create(strUrl);
                httpRequest.Method = "GET";
                httpRequest.ContentType = "application/json";
                if(!String.IsNullOrEmpty(acceptHeader))
                {
                        httpRequest.Accept = acceptHeader;
                }
                SetProxyIfNeededForLDS(httpRequest, httpRequest.RequestUri);
                httpRequest.Headers.Add("Authorization", GetAuthHeader(username, password));
        ServicePointManager.ServerCertificateValidationCallback = delegate
        { return true; };
                return httpRequest;
        }
private static void SetProxyIfNeededForLDS(HttpWebRequest request, Uri uri)
        {
                var _proxyUriForLDS = WebRequest.GetSystemWebProxy().GetProxy(uri);
                var _proxyForLDS = new WebProxy(_proxyUriForLDS, true)
        {
                Credentials = CredentialCache.DefaultNetworkCredentials
```

```
        };


                if (_proxyUriForLDS != null && !String.IsNullOrEmpty(_proxyUriForLDS.AbsoluteUri) &&
                !_proxyForLDS.Address.Equals(uri) && !IsLocalHost(_proxyForLDS))
        {
                request.Proxy = _proxyForLDS;
        }
        }
public static bool IsLocalHost(WebProxy proxy)
        {
        if (proxy != null && proxy.Address != null
        && (proxy.Address.Host.Equals("127.0.0.1") || proxy.Address.Host.Equals("localhost",
        StringComparison.CurrentCultureIgnoreCase)))
        return true;
        return false;
        }
public static void UploadToLDS()
{
        string _provider, _feedName, keyscols;
StringBuilder sbData;
var sbHeader = new StringBuilder();
        sbHeader.AppendLine("# Provider: " + _provider); // e.g. LIM
        sbHeader.AppendLine("# Feed: " + _feedName); //feed name,
e.g Client_XLS_Upload
        sbHeader.AppendLine("# Created: " + DateTime.Now);
        sbHeader.AppendLine("Def:" + keyscols); // all keys & columns
including Date in csv e.g. Date, key1*, key2*, col1,col2,
//note you have to put "*" after each key
        sbHeader.Append(sbData); //csv data, if there is no data for a column, put a comma, e.g. 2014-02-
        02,key1,key2,,col2
                string path = "some folder you specify"; //whatever folder you put temporay upload files
                var fileCommonPart = _provider + "_" + _feedName + "_" +
                DateTime.Now.ToString(MIMICConstants.ISO_8601_DATETIME_FORMAT);
                var cdfFileWOPath = fileCommonPart + ".cdf"; //for meta file, use .cmf, there is no "Date" or
                "DateTime" when upload meta data
                //if (!_hasDateCol) cdfFileWOPath = fileCommonPart + ".cmf";
                var cdfFile = Path.Combine(path, cdfFileWOPath);
                using (var outfile = new StreamWriter(cdfFile))
        {
                outfile.Write(sbHeader.ToString());
        }
                var zipFileWOPath = fileCommonPart + ".zip";
                string zipFile = Path.Combine(path, zipFileWOPath);
                using (var zip = new ZipFile()) // Here I use ionic.zip, opensource zip library @
                http://dotnetzip.codeplex.com/,
```

```
        {
                zip.AddFile(cdfFile, "");
                zip.Save(zipFile);
        }
                string email = null, password = null; //your username & password to Marketplace
                var uri = "https://mp.morningstarcommodity.com/lds/" +
                string.Format("providers/{0}/items?feed={1}&id={2}&type=cf", _provider, _feedName,
                zipFileWOPath);
                //uri = uri + "&partial=True"; //add this if you want to do partial change, i.e. only change values for
                columns specified in file without overwritting other columns
                var webClient = new WebClient();
                webClient.Credentials = new NetworkCredential(email, password);
                SetProxyIfNeededForLDS(webClient, new Uri(uri));
                webClient.Headers.Add("Authorization", GetAuthHeader(email, password));
                webClient.UploadDataCompleted += webClient_UploadDataCompleted; byte[] fileBytes =
                File.ReadAllBytes(zipFile);
                webClient.UploadDataAsync(new Uri(uri), "POST", fileBytes);
        }
                void webClient_UploadDataCompleted(object sender, UploadDataCompletedEventArgs e)
        {
                //do something, like tell users if submit fails or succeeds
                if (e.Error == null)
        {
                MessageBox.Show("Upload submitted to Server", "Information", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        else
        {
        MessageBox.Show(e.Error.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Stop);
        }
}
        public static string GetAuthHeader(string username, string password)
{
        return "Basic " + Convert.ToBase64String(Encoding.Default.GetBytes(username + ":" + password));
}
```

Accessing Marketplace API through Python

```
import pandas as pd
import io
import requests
from requests.auth import HTTPBasicAuth
import datetime

mpurl = "https://mp.morningstarcommodity.com/lds/feeds/<feedname>/<options>"
r = requests.get(mpurl, auth=HTTPBasicAuth(username, password))
data = r.text
print(data)



#Examples

#timeseries API call
mpurl =
"https://mp.morningstarcommodity.com/lds/feeds/PJM_Da_Hourly_Lmp/ts?PnodeID=0,1,100,101,1047909,10479
10&cols=totallmp&fromDateTime=2017-04-12T00:00:00.0000000-04:00&toDateTime=2017-04-
12T23:59:00.0000000-04:00"
#feed info
mpurl = "https://mp.morningstarcommodity.com/lds/feeds/CME_NymexFuturesFIXML"
#ts for a key and col
mpurl =
"https://mp.morningstarcommodity.com//lds/feeds/CME_CmeFutures_continuous/ts?Contract=ED_001_Month&col
s=Settlement_Price&fromDateTime=2018-07-01&toDateTime=2018-07-10"
#keys with limit
mpurl = "https://mp.morningstarcommodity.com/lds/feeds/PJM_Da_Hourly_Lmp/keys?limit=10"
#mcd feed
mpurl =
"https://mp.morningstarcommodity.com/lds/feeds/CME_CmeFutures_continuous/md?Contract=ED_001_Month"
#range for ts
mpurl = "https://mp.morningstarcommodity.com/lds/feeds/ECB_EuroFxRef/ts/range?Currency=USD"
#ts changes
mpurl =
"https://mp.morningstarcommodity.com/lds/feeds/Ice_eurofutures/ts/changes?symbol=BRN6Z&fromDateTime=20
16-10-19&insertTimes=true"
```

Accessing Marketplace Curve API through Python

```
import pandas as pd
import io
import requests
from requests.auth import HTTPBasicAuth
import json


##curve api call example- feed,root,date can be changed as seen fit.(Note: only one date can be specified at a time)
mpurl = "https://mp.morningstarcommodity.com/lds/feeds/CME_NymexFutures/curve?root=NG&cols=Settlement_Price&date=2018-08-28"
#change the username and password here - must be in quotes
r = requests.get(mpurl, auth=HTTPBasicAuth("username", "password"))
#clean up
data = r.text
data
#data in csv format
payload = json.loads(data)
print(payload)
```

Accessing Marketplace Curve API through MATLAB

```
%set options
>> options = weboptions;

%Marketplace Credentials
>> options.Username = '<username>';
>> options.Password = '<password>';

%Content_type is different with different API calls (see API guide)
>> options.ContentType = 'json';

%Usually all the MP APIs are GET (expect publisher worker)
>> options.RequestMethod = 'Get';

%%curve api call example- feed,root,date can be changed as seen fit.(Note: only one date can be specified at a time)
>>mpurl = "https://mp.morningstarcommodity.com/lds/feeds/CME_NymexFutures/curve?root=NG&cols=Settlement_Price&date=2018-08-28";

%pulls back the response in the specified Content_type
>> response = webread(URL, options)
```

Accessing Marketplace Curve API through R

```
#install.packages("RCurl")
library(RCurl)

##curve api call example- feed,root,date can be changed as seen fit.(Note: only one date can be specified at a time)
mpurl <-
"https://mp.morningstarcommodity.com/lds/feeds/CME_NymexFutures/curve?root=NG&cols=Settlement_Price&date=2018-08-28"

#username and password should be separated by a colon(:)
userpw <- "r-client@great_company.com:Secret"

#function call
curve.fit <- function() {
  URL = mpurl
  payload =  getURL( url = URL, userpw= userpw, ssl.verifypeer = FALSE)
  response = read.csv(textConnection(payload), header = TRUE, stringsAsFactors = FALSE)
  response
  }

#output on the consol
curve.fit()

#write to a csv
write.csv(curve.fit(),file = "curve.csv")
```